Fair Multi-Label Reconstruction from Cross-Sections

Research Thesis

In Partial Fulfillment of the
Requirements for the
Degree of Master of Science in Computer Sciences

Raeda Naamnieh

November 2013
Submitted to the Senate of the Technion - Israel Institute of Technology
Tishrei, 5774 HAIFA September 2013

The Research Thesis Was Done Under The Supervision of Prof.
Gill Barequet
in the Faculty of Computer Science

Acknowledgements

The Generous Financial Help of the Technion, and of the Ministry of Science, Technology, and Space, are Gratefully Acknowledged

Contents

List of Figures

Abstract

In this thesis we propose an algorithm for reconstructing a multi-label object from cross-sections in a fair manner. We handle the problem in its full generality: Cross-sections need not be parallel nor complete, every section may contain an unlimited number of contours with any geometries and in any level of containment hierarchy. We focus on the simultaneous reconstruction of an object from contours with multiple labels ("colors"), in scenarios in which interpolating separately between contours of each color results in conflicting (intersecting) reconstructions. We suggest a flexible scheme for combining all the individual reconstructions and resolving fairly the conflicts between them.

Chapter 1

Background

1.1 Introduction

The problem of reconstructing the boundary of a solid object from a series of parallel planar cross-sections has attracted much attention in the literature during the past 35 years. The main motivation for this problem comes from medical imaging applications, where cross-sections of human organs, including all kinds of tissues, are obtained by CT (Computed Tomography), MRI (Magnetic Resonance Imaging), etc. These cross-sections are the basis for interpolating the boundary surface of the organ. The interpolated object can then be displayed in graphics applications or even manufactured by an NC (Numerically Controlled) or an RP (Rapid Prototyping) machine. Another motivation for this problem is the non-destructive digitization of objects: after an object is scanned by an echo-graphic or an X-ray apparatus, the obtained slices are used for the reconstruction of the original object. Yet another motivation is the reconstruction of a 3-dimensional model of a terrain from topographic elevation contours.

Many solutions were suggested for the pure raster interpolation. These usually handle two raster images, where each pixel is either white or black, or assigned a gray-level taken from a fixed range. The interpolation produces one or more intermediate raster images, which smoothly and locally turn the first image into

the second one. Then, the bounding surface is detected using other methods, such as edge detection techniques, for identifying places of transition from the inside to the outside of the object. In the grey level case, these methods include some thresholding mechanism that decides which levels are 'inside' the object and which are not. Cline et al. [CLLC88, LC87] attempted to convert the voxel data directly into a polyhedral surface, suggesting the *marching cubes* technique, which produced very small triangles whose size was roughly the same as that of the input voxels.

Many other solutions assume that the interpolation is preceded by an edge-detection process, which is invoked for each of the slices. Thus, each slice is then considered to be represented by a hierarchy of non-crossing contours, each being a closed simple Jordan curve, which represent the boundaries between "material" and "non-material" areas; in general, the depth and breadth of this hierarchy is not restricted, and a contour may enclose any number of other contours, which themselves may enclose other contours, and so on. In practice, each contour is given as a discrete circular sequence of points along it, and we can thus regard it as a simple closed polygonal curve, whose vertices are the given points. Finally, we may also assume that the exterior, unbounded region in each planar slice represents "non-material" (the model is assumed to be bounded).

Thus, the problem is: Given a collection of planar cross-sections of an unknown object, each consisting of a set of non-crossing, but possibly nested, closed and simple polygonal curves, we want to reconstruct a polyhedral solid model whose cross sections along the given planes coincide with the input slices. A popular simplification of the problem, done in most works that handle collections of *parallel* cross-sections, is to consider only a single pair of successive parallel slices, and to construct a solid model within the layer delimited by the planes of the slices, which interpolates between the given slices. The union (or, rather, concatenation) of these models will give us a solution model for the full problem.

There has been intensive research on this problem in several "waves." We provide here only a brief survey of this history of research. Most of the earlier works

only studied the variant where all the cross-sections are parallel, and each one of them contains only one contour. These studies either sought a global optimization of some objective function, or used a local tiling-advancing rule, after the tiling starting points at the two contours were somehow determined, e.g., the closest pair of vertices between the contours. Such solutions were proposed the seminal work of Keppel [Ke75] and later in [BPCC81, CCLB80, FKU77, KD88, KSD88, SH81, SP88, WA86, WW93]. Then, a few works [CS78, CP94, EPO91, GD82, MSS91, Sh81, ZJH87] suggested methods for handling simple branching cases.

A new generation of algorithms attempted to handle the reconstruction problem in much more generality, allowing (still parallel) cross-sections to contain any number of contours and not assuming anything about their shape, resemblance, and/or nesting hierarchy. Boissonnat [Bo88] (see also [BG92, Ge93]) opened a new era in the research of this problem, presenting the first algorithm that handles the problem in this generality. He constructed the Delaunay triangulation for each slice, projected one triangulation onto the other, and obtained a collection of tetrahedra, aiming to maximize the sum of their volumes.

Barequet and Sharir [BS96] (see also [BST00, SSBT01]) proposed another algorithm that handles the problem in this generality. The algorithm is composed of two steps. The first step matches *similar* portions of contours (using a *geometric-hashing* curve-matching technique) between every pair of adjacent slices and tiles each pair of matched subcontours by a sequence of adjacent triangles. The second step identifies the remaining *clefts* between contours (unmatched portions of contours) and triangulates them while optimizing some objective function.

Bajaj, Coyle, and Lin [BCL96] presented an algorithm similar in nature to that of [BS96], but also provided a profound theoretical ground to the behavior of their algorithm. They formalize the constraints imposed on the reconstructed surface and prove rigorously the correctness of the reconstruction, that is, that the result is a non-self-intersecting surafce. This is one of the rare algorithms which provide a guaranteed running time. The time complexity of the algorithm is $O(n \log n)$ in

the average case and $O(n^2 \log n)$ in the worst case, where n is the complexity of the input.

Barequet et al. [BGLS04] presented another efficient method for handling the general reconstruction problem. The method is based on computing cells in the overlay of the cross-sections that form the symmetric difference between them. Then, the straight skeletons of the selected cells are computed, and then guide the triangulation of each face of the skeletal cells. Finally, the resulting triangles are lifted up in space to form an interpolating surface. This algorithm is somewhat similar to that of Oliva, Perrin, and Coquillart [OPC96]. The latter algorithm is different in the classification of cells, the triangulation scheme, and the method for assigning heights to intermediate vertices.

The current trend is to remove the requirement that all input cross-sections be parallel and that all contours represent a single type of material (tissue in medical data). Payne and Toga [PT94], Dance and Prager [DP97], and Bogush et al. [BTS04] allowed cross-sections to be aligned in a few directions. Boissonnat and Memari [BM07], and Liu et al. [LBD+08] allowed cross-sections to be oriented completely arbitrarily. They compute the arrangement of the planes supporting the cross-sections, then subdivide each cell of the arrangement by using its medial axis, project on this axis the portions of contours lying on the boundary of the cell, match portions of the images of this projection, and reconstruct portions of surface connecting matching portions of the original contours.

Only very few works dealt with the "multi-colored" case, in which the cross-sections simultaneously describe several types of tissues (e.g., muscles, fat, bone, blood, etc.), so that each contour is given a *label* that characterizes its type. For such input, separate reconstructions based on each type of contours regardless of the other types might result in inconsistencies and intersections between the reconstructed surfaces. The methods of Ju et al. [JWC+05] and Liu et al. [LBD+08] are among the few that support multi-colored cross-sections. Edwards and Bajaj [EB11] handle the case in which contours of various colors are given in parallel

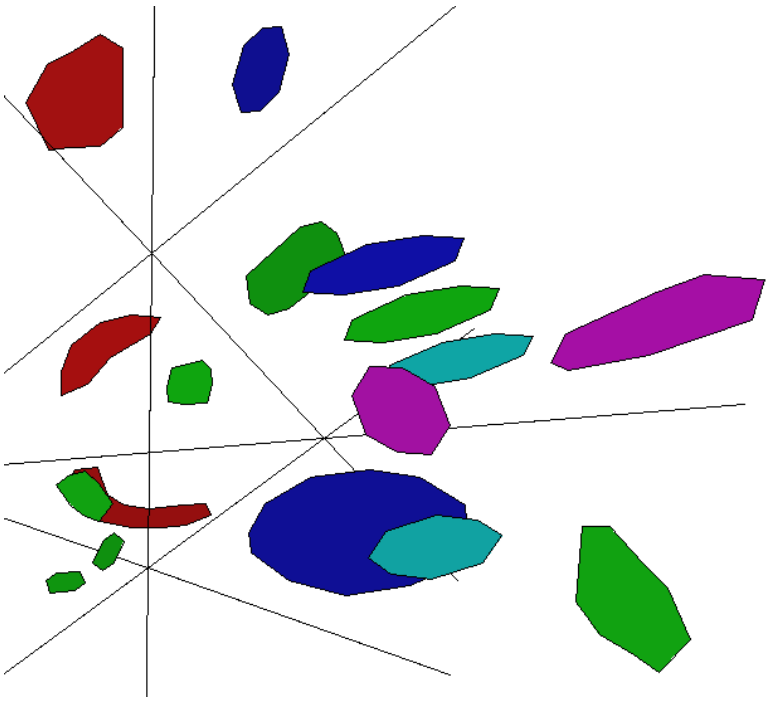
cross-sections, however, the contours intersect *within* the sections. Such inconsistencies are handled by a so-called "intersection removal" algorithm. Bermano et al. [BVG11] describe an *on-line* algorithm for reconstructing simultaneously multi-colored object from arbitrarily-oriented, possibly partial, cross-sections. The algorithm produces smooth results but is very slow in practice.

An additional desired feature is "partially defined" cross-sections, that is, sections in which portions have the "unknown" label, resulting from areas scanned improperly or not scanned at all. Such a situation cannot be handled by all the reconstruction algorithms described so far since in this case the cells of the arrangement of the (portions of) planes are not convex any more. An algorithm by Barequet and Vaxman [BV09] combines all the above-mentioned features plus the ability to handle partially-defined sections. It employs the method of [BEGV08] for computing the straight skeleton of nonconvex cells.

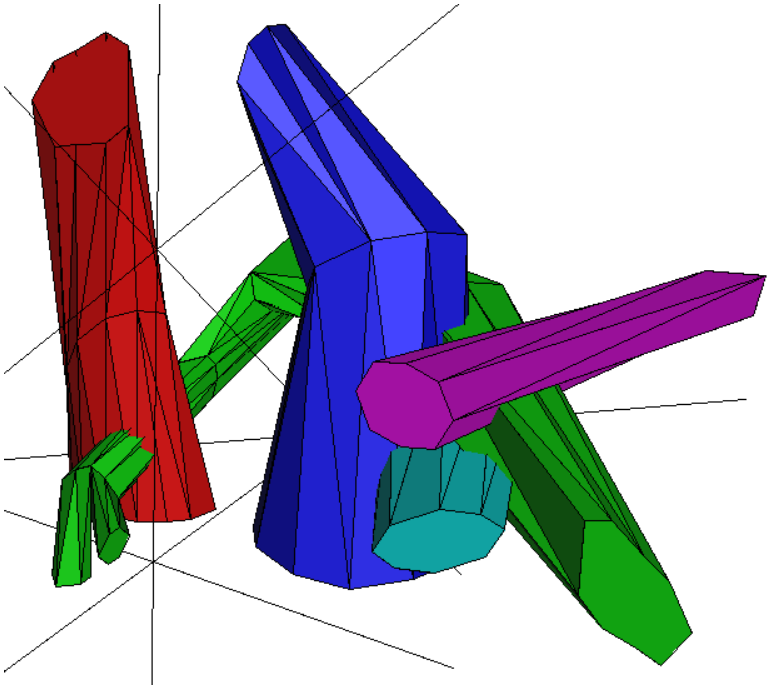
All existing algorithms which reconstruct a multi-labeled object from its planar cross-sections [BV09, LBD+08, BVG11] reconstruct all materials (including the "air") simultaneously while avoiding intersections during the reconstruction. In contrast, we suggest to reconstruct the different materials separately. This provides more flexibility and fewer constraints, and, thus, better and more reliable results. However, intersections between the different materials may appear. Therefore, we suggest a novel scheme for splitting fairly the regions of intersection between the involved materials.

1.2 Statement of the Problem

We are given a set of arbitrarily-oriented partial planar cross-sections of an unknown object O. Each cross-section contains a set of closed labeled contours. The contours specify the intersections of the planes with the object, while the labels represent the material types within these intersections. The goal is to reconstruct O as a labeled triangulated mesh (or meshes), representing the boundaries of the different materials of the object. A typical example is shown in Figure 1.1. The



(a) Input



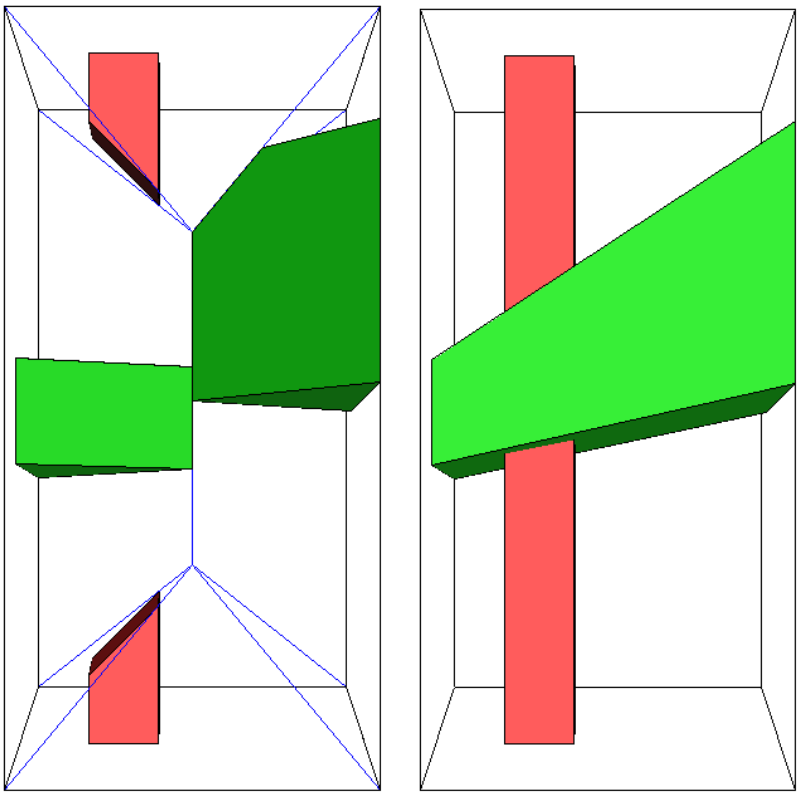
(b) Output

FIGURE 1.1: Typical input and output in a multi-label reconstruction

input is shown in Figure 1.1(a) (the cross-section planes are not displayed, but only their intersection lines), and a possible output is shown in Figure 1.1(b).

As detailed in the introduction, very few published algorithms attempt to interpolate simultaneously between several types of polygons, so-called "labeled" or "colored" contours. An inherent problems of all these algorithms is the "coordination" between the different reconstructions. A simple approach would be to interpolate each type of material separately, and then merge all interpolated volumes into one data set. This approach is prone, however, to intersections between the different interpolations. Ideally, the coordination between them should be an integral part of the algorithm, handling each type of contours while being aware of the other types.

Let us exemplify the problem through a simple but typical example. All existing algorithms, that handle nonparallel cross-sections with multi-labeled contours, first compute the arrangement of planes supporting the sections, and then interpolate separately within each cell of the arrangement. Figure 1.2 shows a simple synthetic example, in which a cell of the arrangement is shaped like a box, having contours of one type (red in the figure) on the top and bottom faces of the cell, and contours of another type (green) on the left and right of the cell. Figure 1.2(a) shows typical reconstruction produced by the algorithms suggested in [BV09, LBD+08]. Both algorithms compute the three-dimensional straight skeleton of the cell, project the contours on the facets of the skeleton, and reconstruct separately within each skeletal cell. This implies that the non-material regions (the "air") is weighted equally to the real material types. The algorithm of [BVG11] is on-line in nature. It defines spatial functions based on the input contours and looks for the zero level set of the functions. Therefore, it generates output which is similar to that of both previous algorithms. A more natural reconstruction would be the one shown in Figure 1.2(b), accompanied by a fairing mechanism to handle the region of intersection between the reconstructions of the two material types. Obviously, the problem becomes much more complicated when there are many contours of each type, with more complex shapes and topologies, and especially when there are more



(a) Erroneous (b) Desired

FIGURE 1.2: A typical problem in reconstructing from multi-label contours

than two types of material, making the interactions between the reconstructed surfaces much more complex and involved.

The synthetic example above also shows a problem inherent to all skeleton-based reconstruction algorithms: The introduction of skeletal faces, which results in subdividing every cell of the arrangement of planes (supporting cross-sections) into skeletal cells, disconnects contours of the same type that should be interpolated together. This is seen clearly in Figure 1.2(a), in which the two red contours lie in two disconnected skeletal cells. In addition, the reconstruction of the red material is cut in an unnatural way that depends on the location of the red contours relative to the skeleton. The same figure also shows an unnatural connection between the green contours.

Chapter 2

The Algorithm

2.1 Basic Solution

Our strategy for dealing with "contradictions" caused by intersecting reconstructions of multiple types of material will be to interpolate each type separately, then apply a fairing mechanism for splitting the intersection volume(s) between the competing types. For example, the small volume in Figure 1.2(b), that is the intersection between the red and the green reconstructions, will be either associated completely to one of the colors, or split between them according to some scheme. To this aim we suggest a set of splitting and fairing rules, some of which are logical, quantitative, or geometric.

The algorithm proceeds as follows:

1. Construct the arrangement of the planes supporting the cross-sections.
2. Compute the subreconstruction for each label in each cell of the arrangement.
3. For each cell of the arrangement:
   (a) Subtract the regions of conflict (intersections) from the subreconstructed meshes. Also subtract the corresponding faces (or partial faces).
   (b) For each connected component of the intersection regions:

i. Set weights and labels to its faces.

ii. Compute the weighted straight skeleton.

iii. Identify the two labels corresponding to each skeletal face, and select the faces with two different labels.

iv. Triangulate the selected skeletal faces and associate them to the appropriate subreconstructions.

(c) Join the triangles from Steps 3(a) and 3(b.iv).

4. Join all the generated triangles into closed volumes.

2.2 Details of the Algorithm

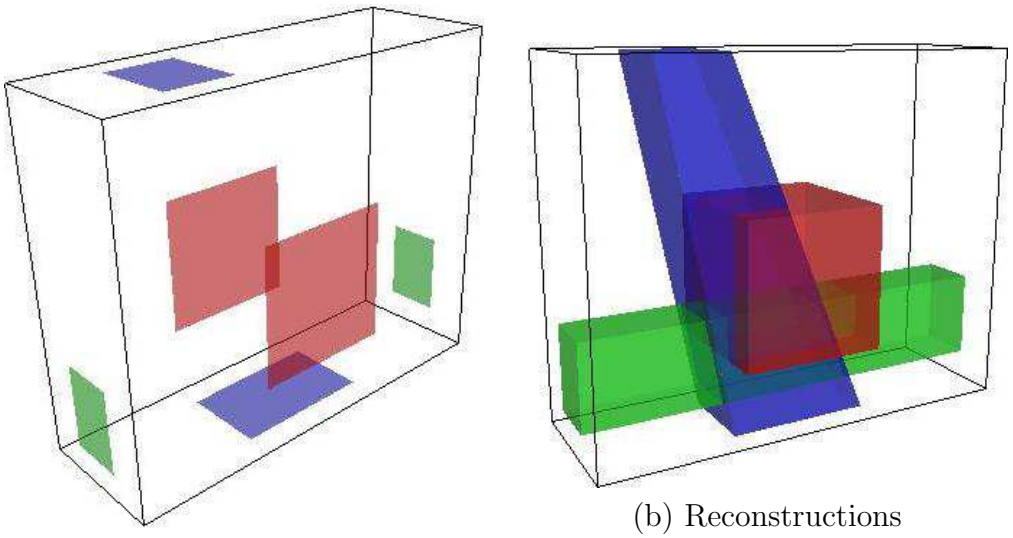
In this section we provide more details of the algorithm. Steps 1 and 2 are standard and are not specific to our algorithm. For Step 1 we complete partial planes and then compute their arrangement as in [BVG11]. For Step 2 we take any reconstruction algorithm available for nonparallel planes, e.g., those of [BM07, BTS04, DP97, PT94]. Figure 2.1(a) shows a sample cell with contours of three colors, input to one application of the algorithm, while Figure 2.1(b) shows the three individual subreconstructions of each colored material separately.

2.2.1 Fairing rules

The user may apply several fairing rules in order to resolve intersections between the reconstructions of the different material regions. Our system allows the user to set static rules in advance, or to fix them interactively during the course of the algorithm. In the latter case, the user may apply different rules to different cells of the arrangement of planes.

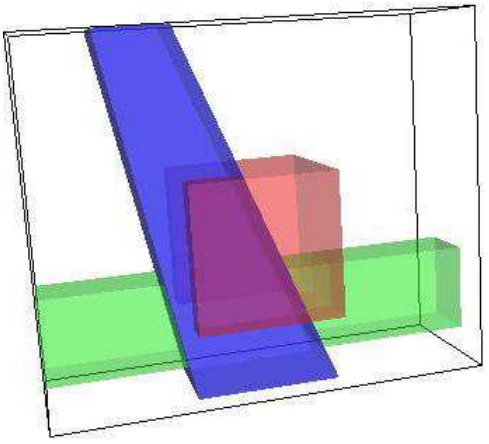
Possible fairing rules include the following:

1. Material of a specific type can stay as-is, that is, annex all regions of intersections with other types of material. See Figure 2.1(c) for an example.



(a) Labeled contours

(b) Reconstructions



(c) Blue remains as-is

FIGURE 2.1: Typical input to the algorithm

2. A special treatment can be assigned to *tunnels*, that is, volumes surrounded by another non-zero-genus volume. Again, tunnels can be chosen to stay as-is (irrespective of which material they are in), or assigned weights (see Rule 4) that are different from their regular weight.

3. Optionally, material types can be prioritized in order to resolve ties in cases where two or more intersection materials (or tunnels, due to the previous rule) are chosen to stay as-is. If ties are not resolved by priorities, then their intersection region(s) are subject to the last fairing rule.

4. Materials of different types can be associated with *weights* to be used for determining the split of an intersection region between the involved materials.

(By default, all material types are associated with equal weights.)

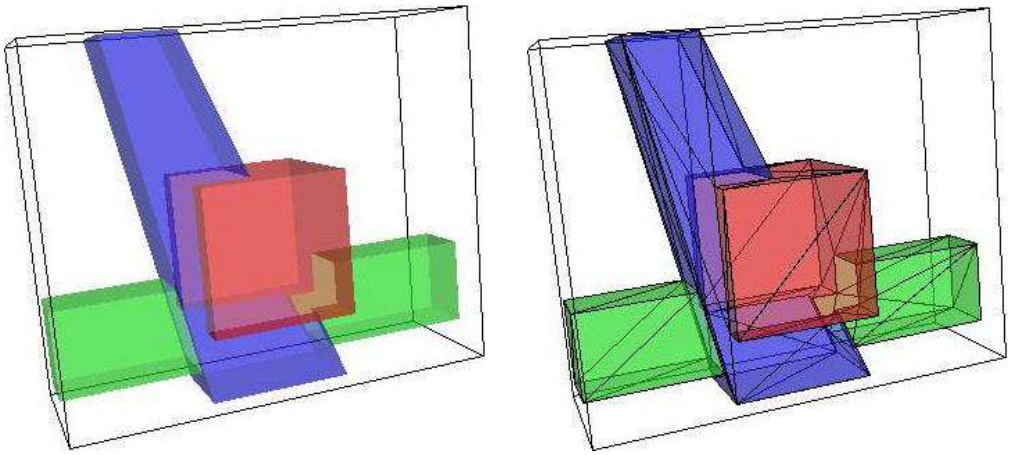
While all rules are geometric in nature, aiming at splitting regions of intersection fairly between the involved material types, some of the rules also have a topological flavor: Imagine a blood vessel going through a muscle; one would not want the blood vessel to break into two disconnected pipe-like pieces, in which case Rule 2 would be applied. As another example, a bone is stiffer than a muscle that is intersecting it, therefore giving the bone a weight higher than that of the muscle will almost surely keep the original shape of the bone while the muscle will bend around the bone. Therefore, Rule 4 would be applied in this case.

We could define general tools that are global in the sense of sharing information between different cells. For example, a material which is involved in a high number of intersections in different cells of the arrangement may be associated with a low weight or may be reconstructed with another algorithm. This is one of the very few times in the history of algorithms suggested to solve this problem, in which a global treatment within each cell is considered.

2.2.2 Subtracting regions

In case a material is to remain as-is, and this requirement is not challenged by an intersection with another material with the same property (and with the same priority), all intersections of this material with other materials are subtracted from the other materials and annexed to the former material. Figure 2.1(c) shows the reconstruction in this case when the blue material remains as-is. This example is less interesting and is, thus, not worked out here.

On the other hand, intersection regions that are not resolved by the simple priority rules, are subtracted from the involved materials and become subject to the last fairing rule, which is the main step of the algorithm. All the intersection regions (of two or more of the subreconstructions) are subtracted, and then united into one or more connected volumes that should be split fairly between the subreconstructions. Figure 2.2(a) shows, for example, three intersecting subreconstructions (red, blue,



(a) Subtraction of volumes (b) Triangulated regions

FIGURE 2.2: Subtracting conflict areas and triangulation

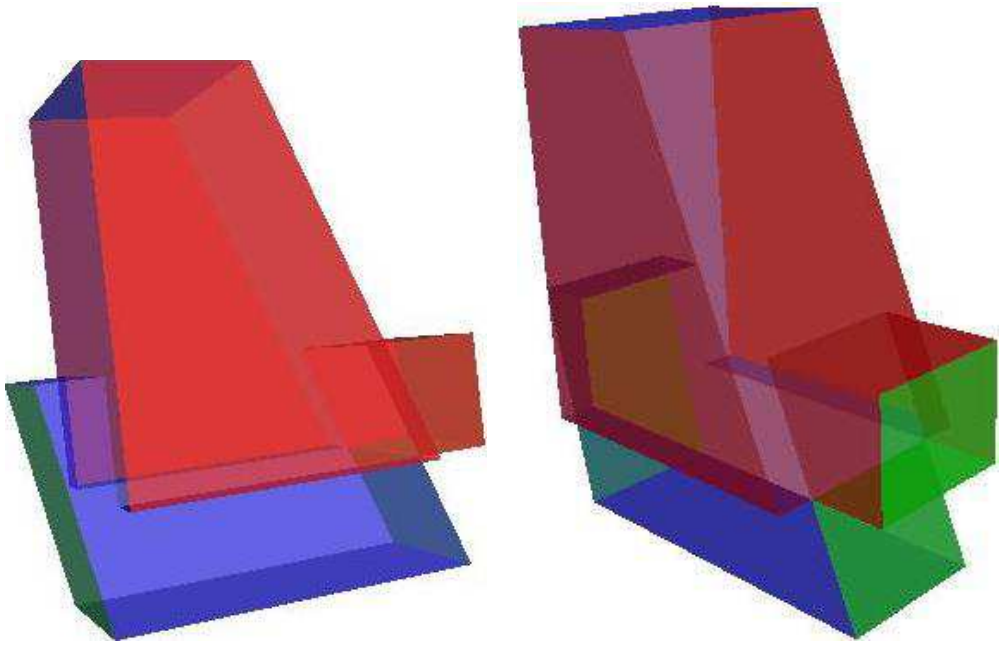
and green) after subtracting the intersection region. Since the boundary of an intersection of volumes, bounded by triangulated meshes, may consist of more complex faces, we triangulate all the newly created faces (see Figure 2.2(b)).

Our implementation takes special care to orient all the triangles appropriately (so that every volume will be described by a properly-oriented boundary) and to match them correctly (so that the interface between neighboring volumes will consist of oppositely-oriented, but otherwise identical triangles).

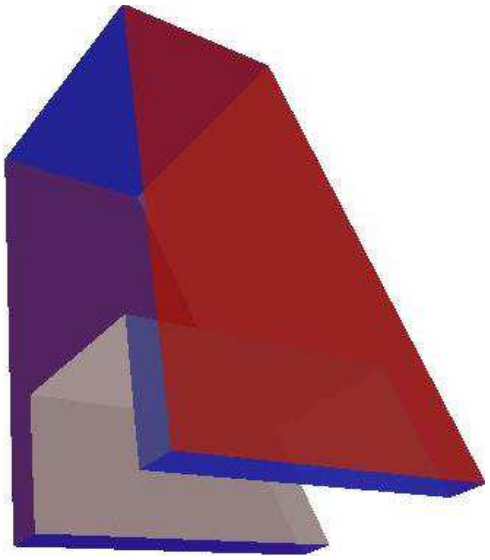
2.2.3 Processing intersection regions

This is the main ingredient of the algorithm. The goal of this step is to divide fairly the intersection regions (or volumes) into portions, associate an appropriate label to each portion, and, finally, to merge these portions with the original subreconstructions. Needless to say, each portion of the conflict region(s) should be connected to the subreconstruction from which it originated.

First, every face of an intersection region is labeled by the (unique) material that contributed it. Figure 2.3(a) shows the intersection region of the three colored materials corresponding to the subreconstructions seen in Figure 2.1(b). Figure 2.3(b) shows another view of this intersection. Figure 2.3(c) shows the intersection region that would be created between the red and the blue materials if we took the green



(a) Three subreconstructions (b) A different view of (a)

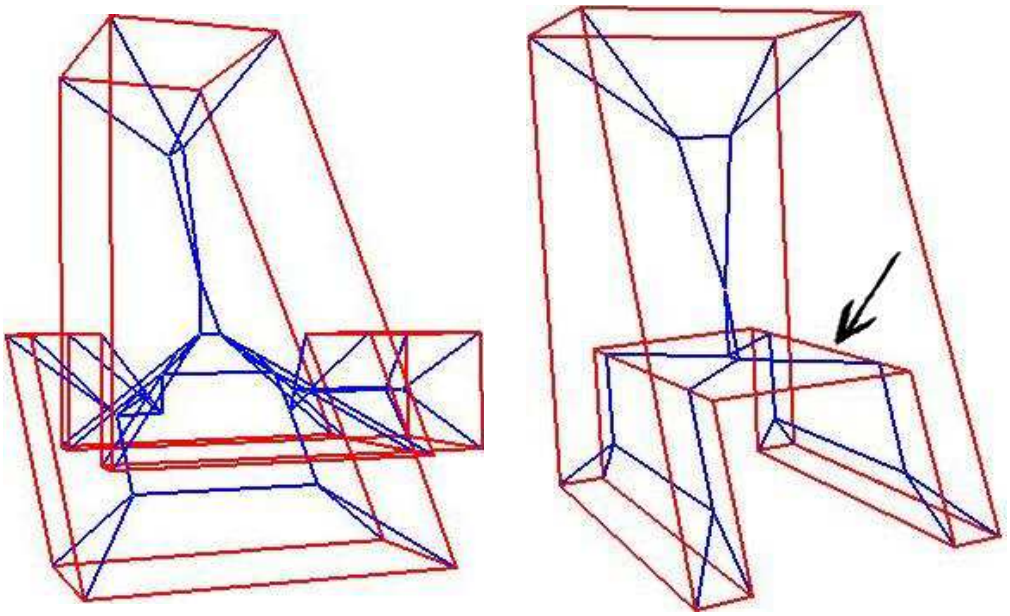


(c) Two subreconstructions

FIGURE 2.3: Coloring the faces of the conflict (intersection) regions

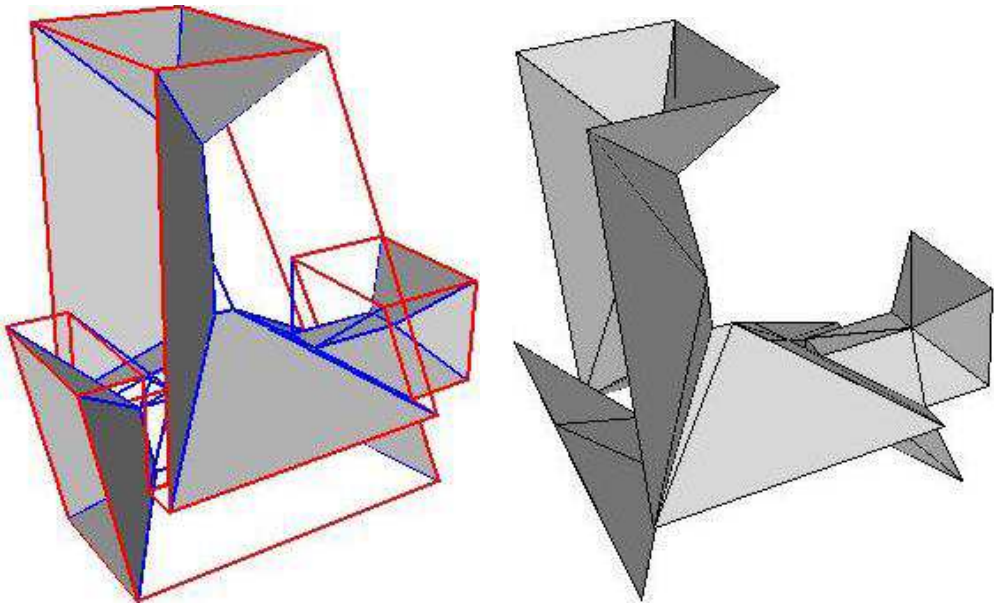
material as-is, and, hence, subtract it from the conflict region. The weight of the face is set according to the corresponding material and the applied rule.

Second, the straight skeleton of the 3-dimensional intersection region is computed by using the algorithm of [BEGV08]. We use a weighted version of the algorithm, where the weight of the faces of the region are set as explained above. In practice, this means that every face "moves" in a different speed; a higher velocity of a face implies that the face grabs more swept volume of the interior of the intersection



(a) Three subreconstructions (b) Two subreconstructions

FIGURE 2.4: Straight skeletons of intersection regions



(a) Before triangulation (b) After triangulation

FIGURE 2.5: Skeletal faces and their triangulations

region. Thus, the higher the weight of a face is, the bigger the volume of the respective portion of the intersection region that it grabs is. Figures 2.4(a,b) show the straight skeletons of the two intersection regions displayed in Figures 2.3(a,c), respectively.

Now, every skeletal cell is defined by exactly one labeled face of the intersection

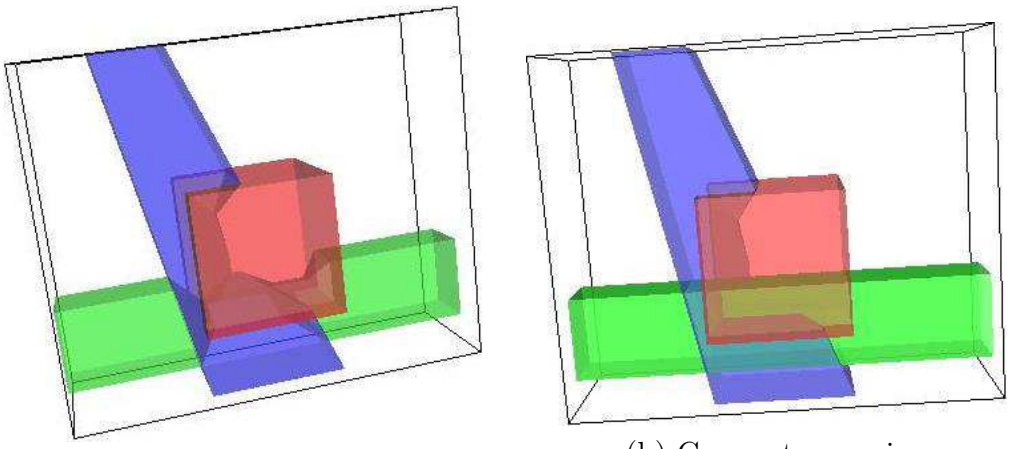
region. Moreover, each skeletal face is shared by exactly two neighboring skeletal cells, so as a third step we mark each skeletal face with the pair of labels of the defining faces of its two neighboring cells. We pick only skeletal faces that are marked with two *different* labels since these will split the conflict region into the different types of material. The skeletal cells are now annexed to the different materials according to their associated labels. The skeletal faces that bound them inside the conflict region are duplicated, in two opposite orientations. The two copies of these faces are added to the boundaries of the materials, respective of the two labels. Consider, for example, Figure 2.5. Figure 2.5(a) shows the selected skeletal faces that correspond to the example shown in Figures 2.3(a) and 2.4(a).

There is one degenerate case in which faces of different subreconstructions, that also contribute to their intersection, overlap or partially overlap in space. In this case the material which contributes a face of the boundary of the intersection region is not defined. To make the algorithm accommodating this case, we do not assign a label to the face, and give it the weight 0. As is shown below, this solves the degenerate case. For example, note the marked face in Figure 2.4(b) (or the white faces in Figure 2.3(c)).

Fourth, the skeletal faces are triangulated in order to obtain triangulated meshes. For example, consider again Figure 2.5. Non triangular faces in Figure 2.5(a) are triangulated, and the result is shown in Figure 2.5(b). The displayed surface will split the conflict region between the three material types.

Finally, all the triangles generated in Steps 3(a) and 3(b.iv) of the algorithm are collected—they form the new boundaries of the different types of material, after splitting fairly all the regions of intersection. Figure 2.6(a) shows the final result of the example worked out above, when all three colors have equal weights. Figure 2.6(b) shows the final result of this example when the green material is chosen to stay as-is, and the red and blue materials have equal weights.

To emphasize the effect of the weights and the intersection shape, we provide a simple example of a conflict region in which we experimented with different weights and subreconstructions. Figure 2.7(a) shows a sample cell of the arrangement



(a) Equal weights (b) Green stays as-is

FIGURE 2.6: Final reconstruction after fair splitting

of cross-sections, with green and blue contours, while Figure 2.7(b) shows their two respective intersecting subreconstructions. Figure 2.7(c) shows different splits between the two colored materials, depending on the ratio between the respective weights. Note the continuous change of the split in this sequence. Figure 2.7(d) shows different splits between the fixed green subreconstruction and a gradually growing blue subreconstruction. In this example, the materials were associated with equal weights. When the relative sizes of the subreconstruction are given (in a particular input), the user may choose the associated weights, and both parameters affect the split of the conflict region. A combination of these given and user-defined factors will lead to a fair final reconstruction.

We conclude this section by proving that the fair-split algorithm preserves the connectivity of the original subreconstructions, unless there are tunnels that are not treated by the appropriate fairing rule.

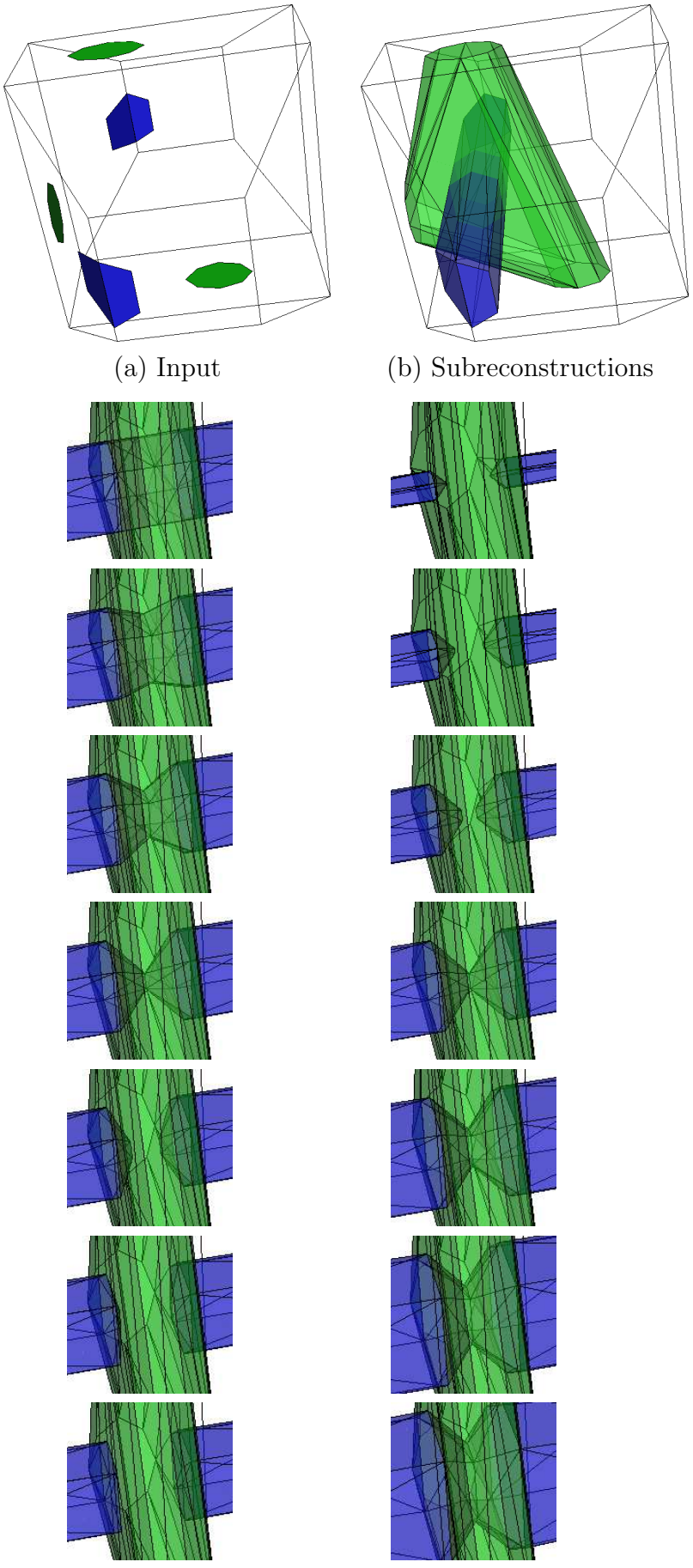
Theorem 2.1. *Labeled portions, created by splitting an intersection region, are connected to the respective subreconstructions.*

Proof. Each boundary face f of a conflict (intersection) region C has a label that was originated by a subreconstruction R with the same label. It is known [BV09] that while computing the 3-dimensional skeleton of C, the volume swept by f is continuous (moreover, it is monotone), and that f is part of its boundary. Thus, the portion of C, respective of f, is precisely the skeletal cell that contains f on

its boundary. Hence, after uniting all such portions with R, the subreconstruction containing f on its boundary (in an opposite orientation), and canceling out these two copies of f, we obtain a connected region.

Note that R can be *broken* into $k > 1$ parts after subtracting C. In this case, there will be k surfaces patches (that originated from R) on the boundary of C, thus, having the same label as R. These k patches will result in k skeletal cells which will further be merged into the respective k parts of R. However, disconnection can only be caused by other subreconstructions whose intersection with R breaks it into more than one part. This means that we have a tunnel (or tunnels), which can be treated simply by the tunnel-fairing rule.

It remains to prove the claim for the degenerate case, in which a face f bounding C is contributed by several subreconstructions. In this case we do not label the face, and assign it the weight (velocity) 0. The effect of this modification is that f does not grab any portion (skeletal cell) of C at all, while some other portions of C extend all the way and touch f. Refer, for example, to R, one of the subreconstructions that contributed f. If no other portion of C, labeled as R, touches f, then R simply loses all its portion of the conflict region to other subreconstructions, and no discontinuity is created. On the other hand, if another portion of C, labeled as R, extends all the way and touches f, this means that locally no discontinuity was introduced either; on the contrary—this part of R now became connected to another part of R, through a skeletal cell swept by another face labeled by the same color as R. □



(a) Input (b) Subreconstructions

(c) Varying weight (d) Varying intersection shape

FIGURE 2.7: The effect of different weights and intersections

Chapter 3

Results

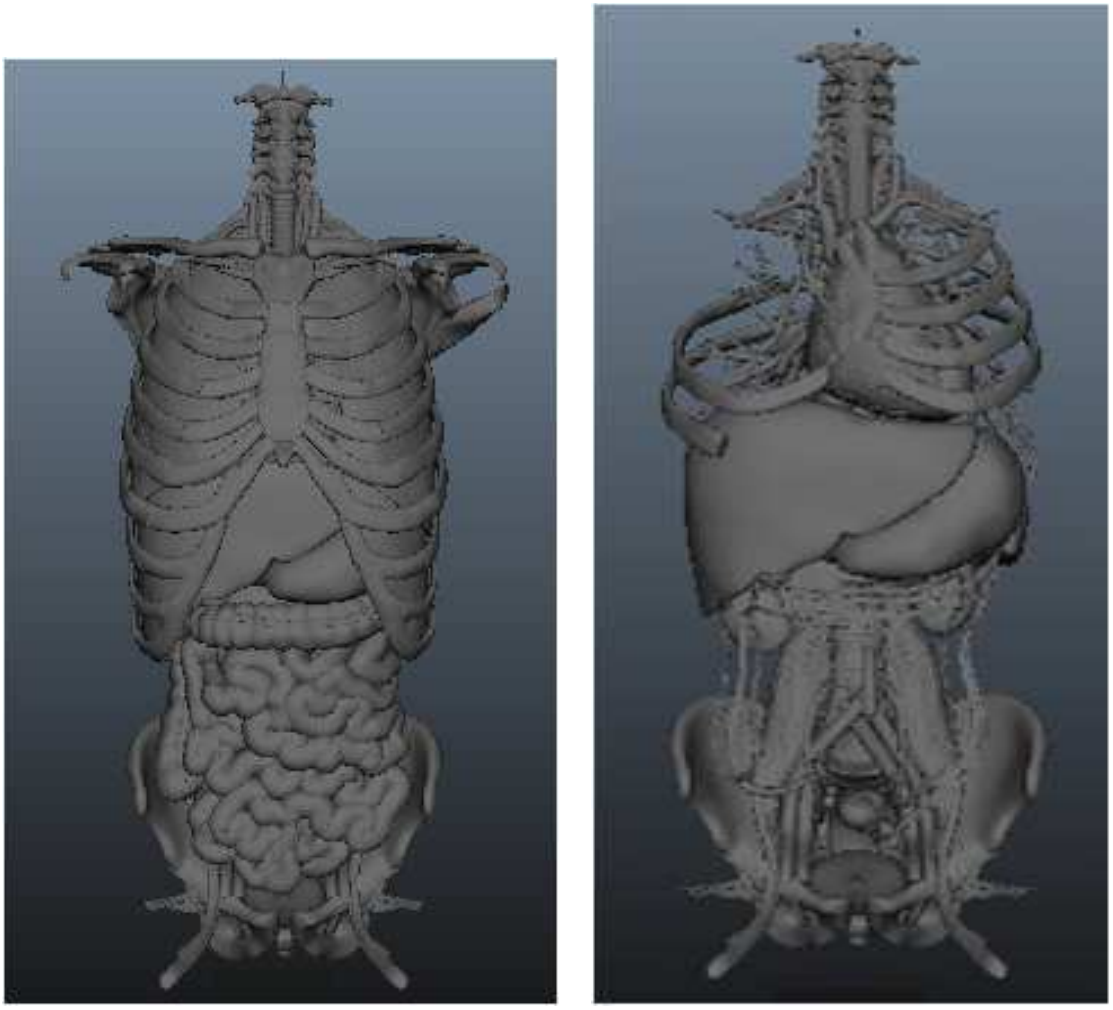
3.1 Implementation

In this section we start by presenting in detail the way in which we created the inputs of the algorithm. Then we follow by describing our implementation of the various reconstruction algorithms and the algorithm for dealing with conflicts.

3.1.1 Input generation

To test our algorithm with real data, we computed cross-sections of a few human organs by cutting existing three-dimensional data. Then, we applied an algorithm similar to that of [BM07] for reconstructing each individual material, ran our fairing algorithm, and finally compared our results with the original organs and with the results of the algorithm in [BV09].

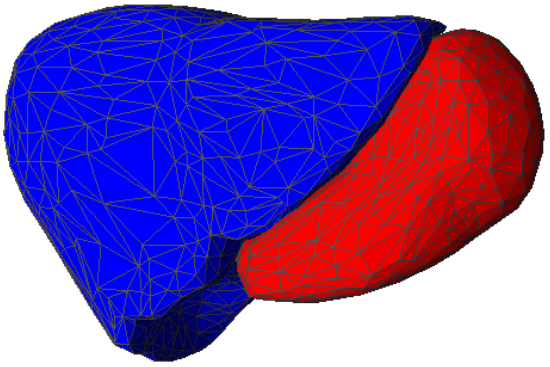
For generating the inputs to our algorithm, we used various human organs, most of which taken from `http://www.zbrushcentral.com/showthread.php?169189` . This database includes a 3D model of a full body male. It has just about every muscle group, every bone and almost all organs. Figure 3.1 shows some internal organs obtained after filtering the database. Since the original scanning was usually done in a high resolution, we simplified the meshes by using the quadric edge-collapse



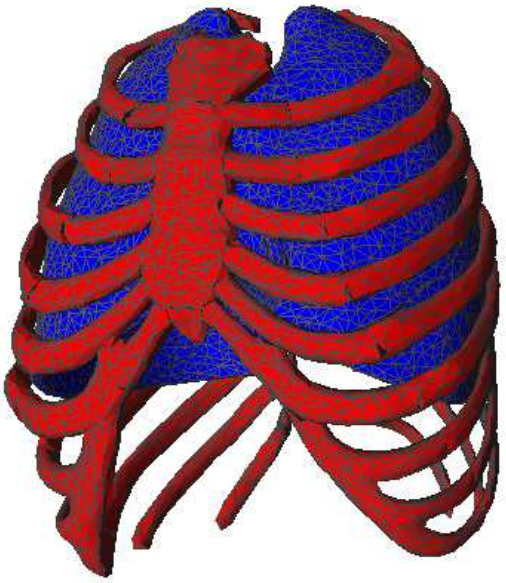
(a) Internal organs (b) Another set of internal organs

FIGURE 3.1: 3D model of a full body male

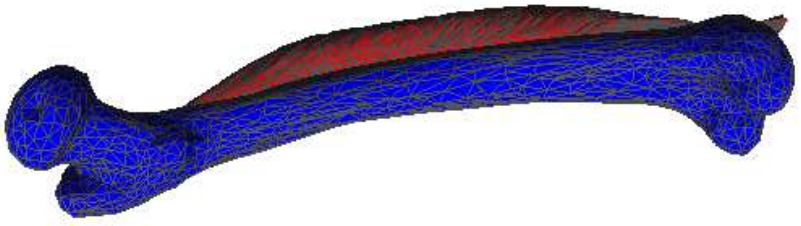
decimation technique to obtain meshes with fewer triangles, which still described clearly the original organs. We then sampled a few organs (see Figure 3.2 and Figure 3.8(a)) and computed a few cross-sections of them. Figure 3.3(a) shows a heart while Figure 3.3(b) shows the contours on one cross-section of it. Figure 3.3(c) is an example of a cross-section of lungs, while Figure 3.3(d) includes three cross-sections of both lungs and a heart, two of which are parallel and the third is orthogonal to them. Figure 3.3(e) includes some parallel cross-sections of both a stomach and a liver (see Figure 3.2(a)), while Figure 3.3(f) contains a few more parallel cross-sections of the same organs. Sometimes these contours intersect. Figure 3.4(a) shows parallel cross-sections of four organs: a rib cage (blue), lungs (green), a heart (red), and a soft tissue that connects the front side



(a) Liver (blue) and stomach (red)

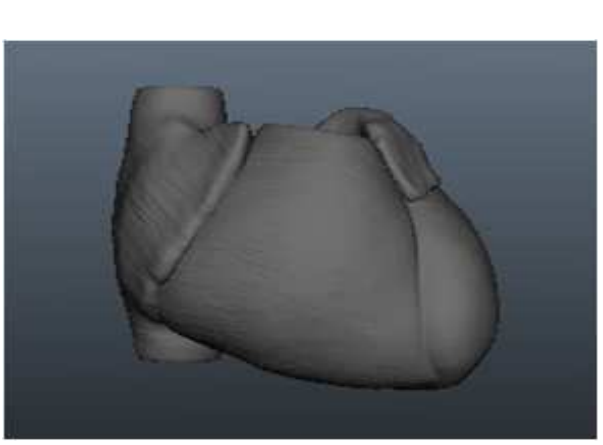


(b) Lungs (blue) and ribs (red)

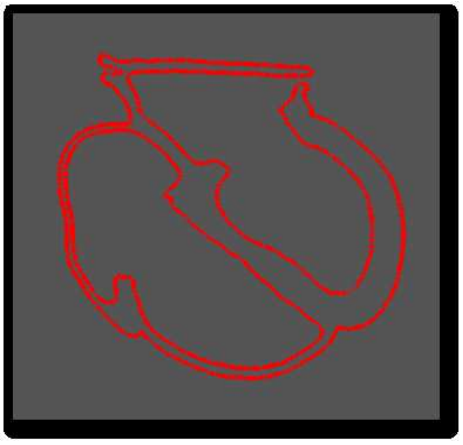


(c) Bone (blue) and muscle (red)

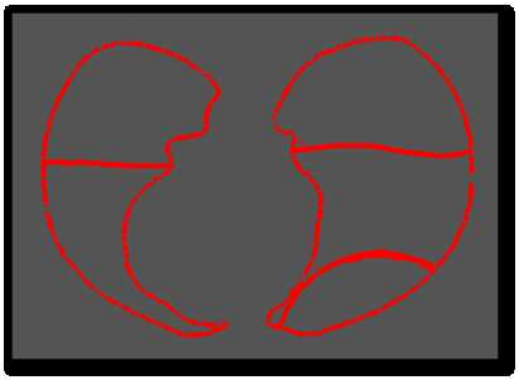
FIGURE 3.2: Full reconstructions of organs



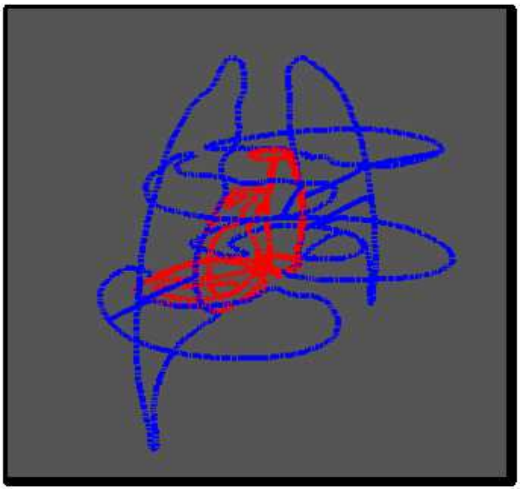
(a) A full heart



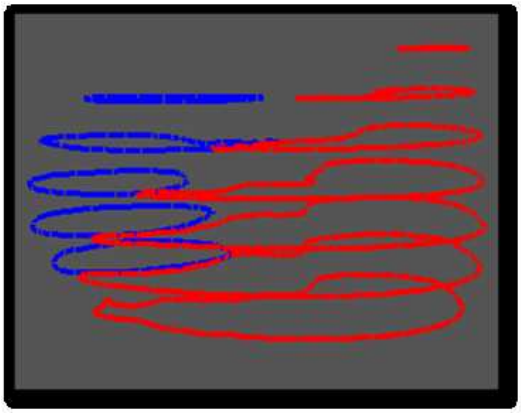
(b) Heart (cross-section)



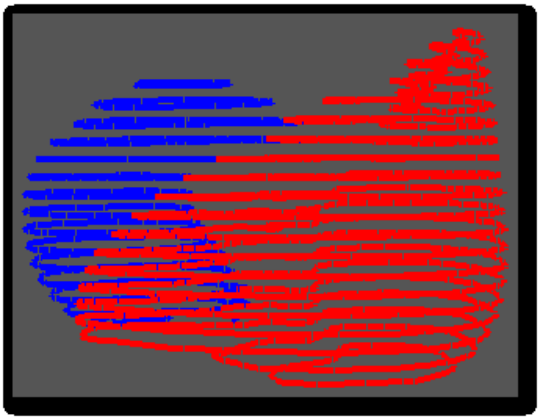
(c) Lungs (cross-section)



(d) Heart and lungs (cross-sections)

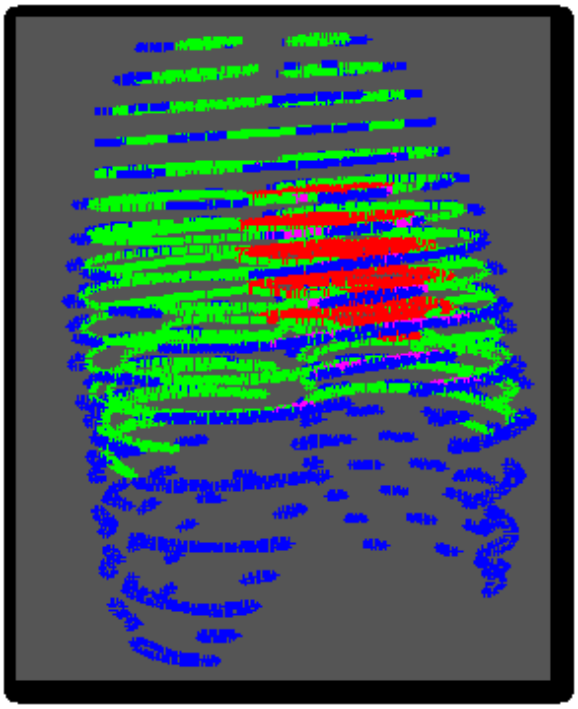


(e) Liver and stomach (cross-sections)

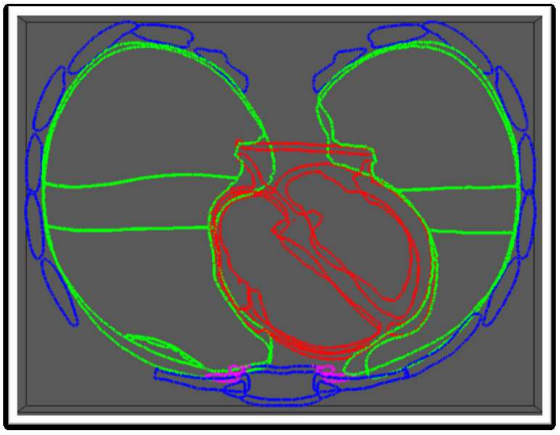


(f) Another set for liver and stomach (cross-sections)

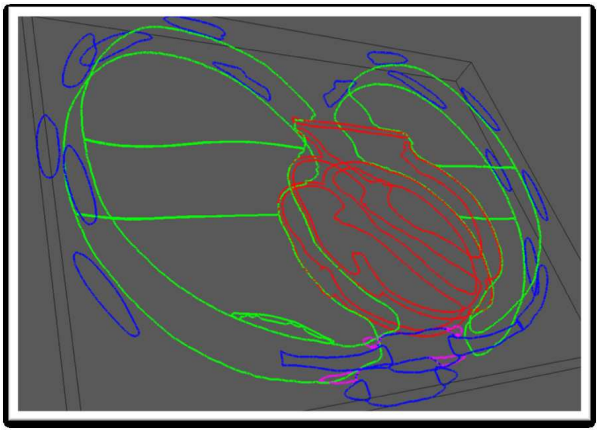
FIGURE 3.3: Cross-sections



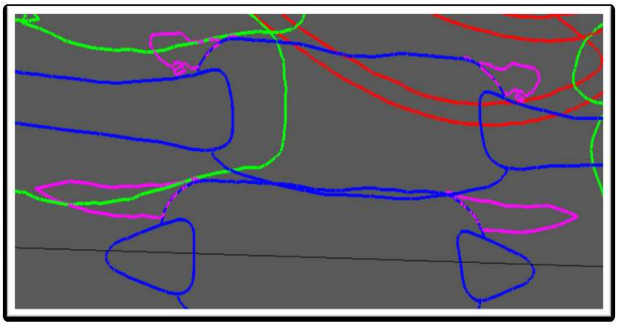
(a) Parallel cross-section of four organs



(b) Two parallel cross-sections



(c) A different view of (b)



(d) Zoom-in into (b)

FIGURE 3.4: Input intersections

of the lungs with the rib cage (magenta). Figures 3.4(b–c) show a close-up of two median cross-sections in different views, and Figure 3.4(d) shows the intersection of the magenta and green contours. We then solved these 2D intersection in a manner similar to that in the 3D case, using the 2D weighted straight skeleton, and finally we merged close vertices of the obtained contours to prevent numeric issues.

3.1.2 Reconstruction

We obtained from Pooran Memari a piece of code which implements the algorithm in [BM07] in order to reconstruct each individual material. We extended it in order to implement our final reconstruction algorithm. We have implemented the reconstruction algorithm of [BV09] in order to compare our results with its results. First, this algorithm computed the arrangement, then, it computed the straight skeleton of each cell of the arrangement, and, finally, it projected the contours orthogonally into the skeleton's faces and connected them.

3.1.3 Intersections

We have used the CGAL library [CGAL] in order to evaluate the boolean operations. The algorithm proceeds as follows:

1. For each cell of the arrangement, compute the connected regions of intersections:
    1.1 Compute intersection volumes between each two materials.
    1.2 Unite the volumes.
2. For each connected region of intersection:
    2.1 Subtract this volume from the involved materials.
    2.2 Split the intersection region between the involved materials.

The algorithm which implements the weighted straight skeleton proceeds as follows:

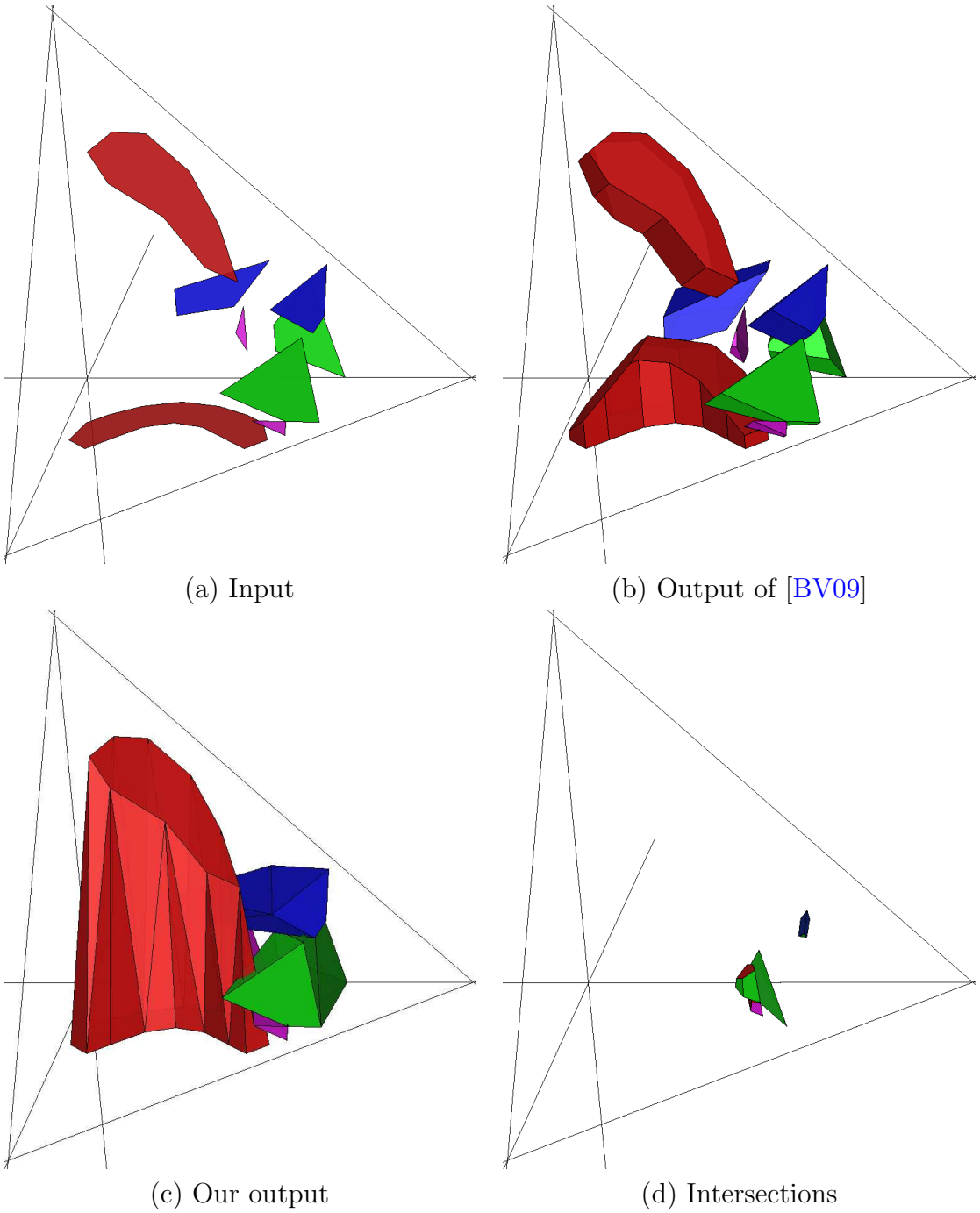
1. Compute the candidate vertices of the straight skeleton.
2. Filter the vertices.
3. Compute the faces of the skeleton.

3.2 Experimental Results

In this section we present results obtained by applying our algorithm on synthetic and real inputs.

3.2.1 Synthetic examples

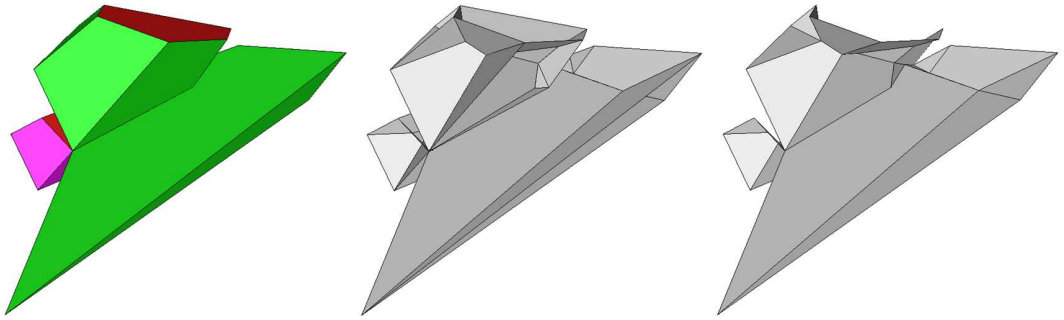
We provide another more complex synthetic example which emphasizes our new ideas. Figure 3.5(a) shows the input to the algorithm. Figure 3.5(b) shows the reconstruction of both algorithms in [BV09] and [LBD+08]. (The algorithm of [BVG11] generates a similar result.) Note the discontinuity of the different materials. We have used the algorithm of [BM07] to reconstruct each individual material; the result is shown in Figure 3.5(c). Figure 3.5(d) shows the generated conflicts, manifest in two little intersection regions. Figures 3.6(a–c) show the first (bigger) intersection in orientations that reflect the "points of view" of the subreconstructions of the involved green, red, and magenta (after subtracting the conflict region). The left figures show the intersection faces and their associated labels. The middle figures show the skeleton faces when equal weights were given to all the materials. Note that each face defines one monotone cell. Finally, the right figures show the selected skeleton faces, that is, faces shared by cells with different labels. Figure 3.6(d) shows the second intersection region. The left figure shows the faces and their associated labels (green and blue), while the right figure shows the intersection with inner selected skeleton faces (shown in gray).



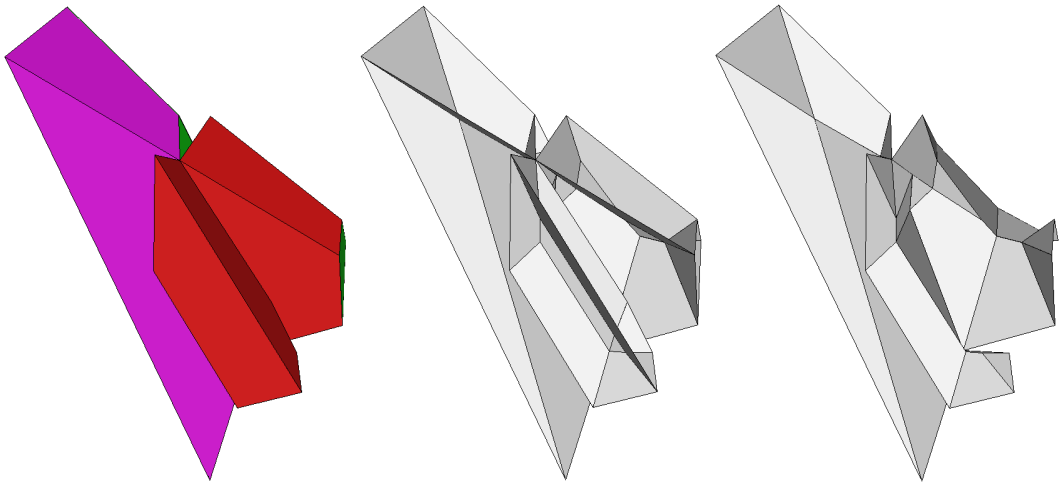
(a) Input (b) Output of [BV09]

(c) Our output (d) Intersections

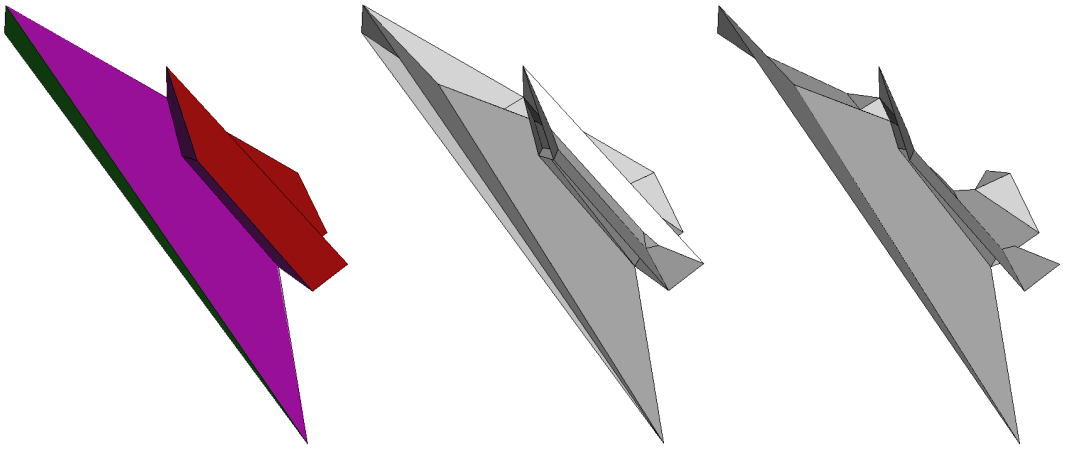
FIGURE 3.5: A synthetic example



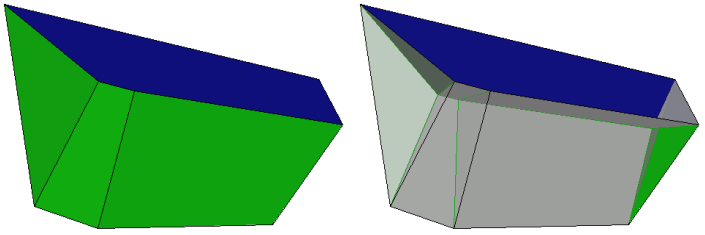
(a) First intersection: green view



(b) First intersection: red view

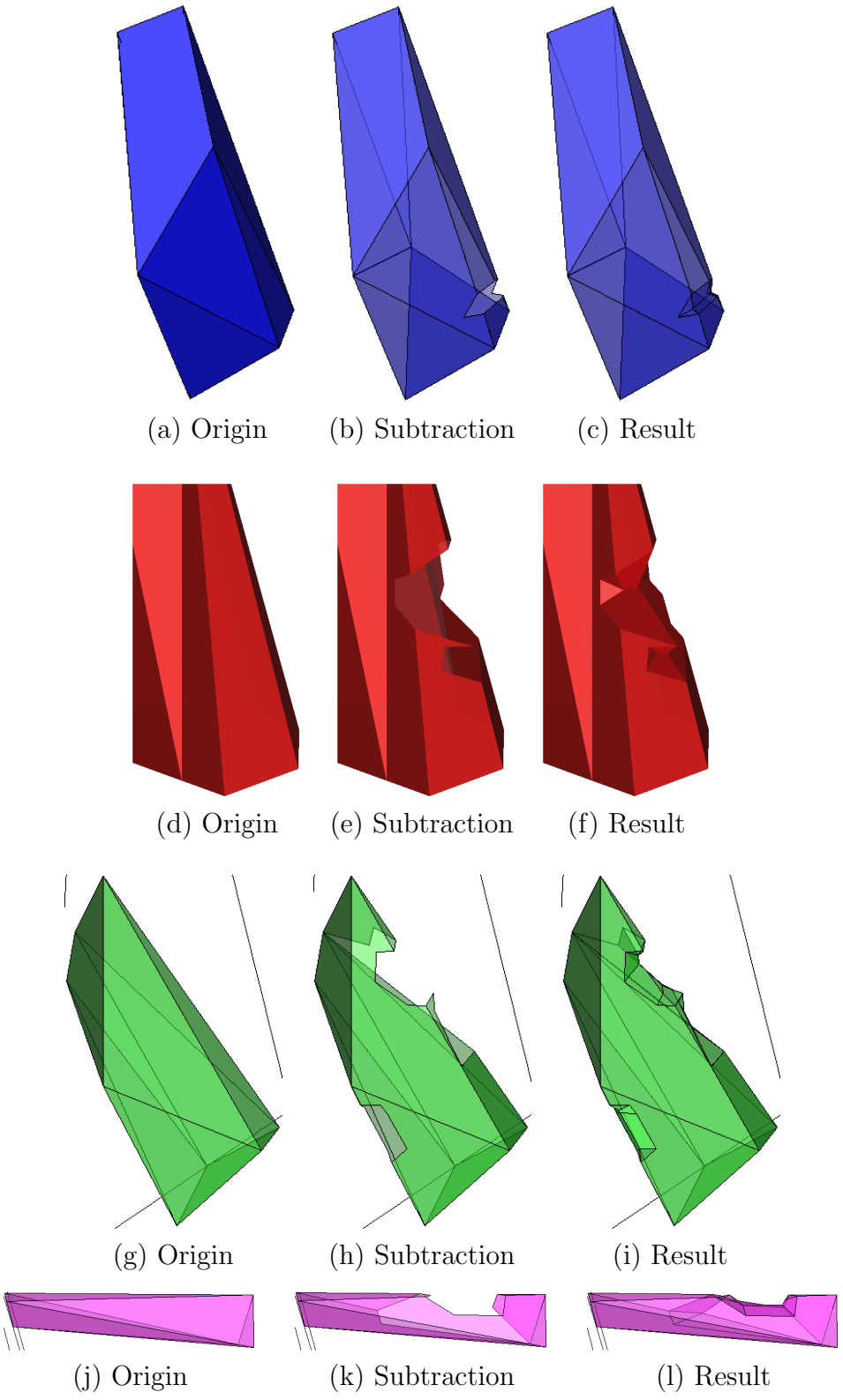


(c) First intersection: magenta view



(d) Second intersection

FIGURE 3.6: Intersection treatment



(a) Origin (b) Subtraction (c) Result

(d) Origin (e) Subtraction (f) Result

(g) Origin (h) Subtraction (i) Result

(j) Origin (k) Subtraction (l) Result

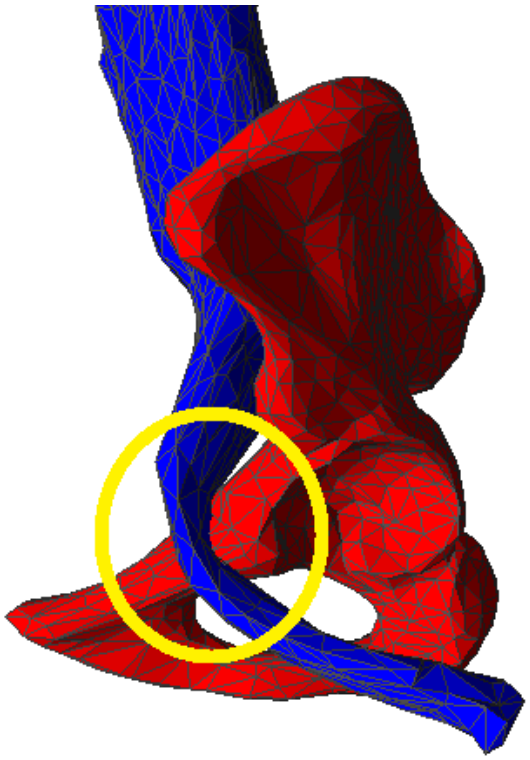
FIGURE 3.7: The modified regions

Figure 3.7 shows the modified regions. For each material, the figure first shows the original shape, then the faces (or partial faces) after subtracting the intersection faces, and finally, the result after combining the selected skeleton faces, where each face shared by two cells is duplicated and combined with both appropriate materials. The final result is quite similar to the one shown in Figure 3.5(c); it was omitted because it is hard to notice the difference between the geometries before and after applying the algorithm.

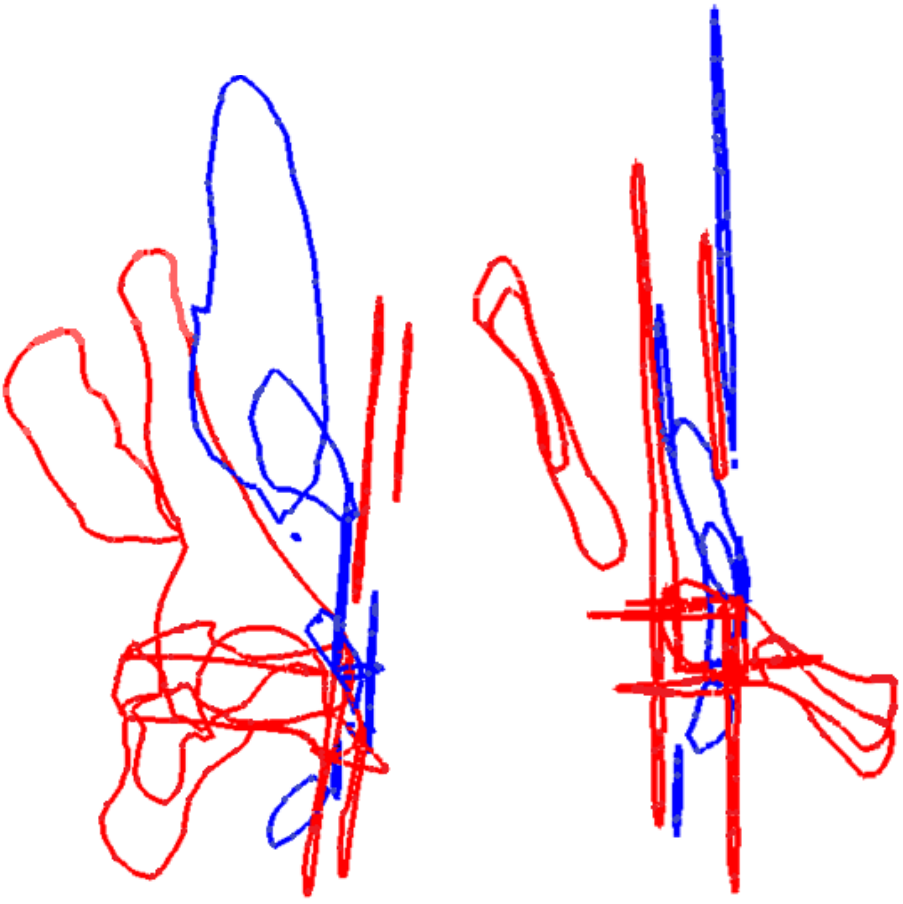
3.2.2 Real examples

We also provide one real example. We generate the input as discussed in the sub-section 3.1.1. Figure 3.8(a) shows two simplified organs, a soft tissue (in blue) curling around part of a pelvis (in red). The yellow ellipse surrounds a region around which we computed six cross-sections. Figures 3.8(b,c) show the labeled contours on these cross-sections. Figure 3.9(a) shows the arrangement cell and the input labeled contours. Figure 3.9(b) shows the inner subreconstructions. The conflict regions are marked with yellow ellipses. Figure 3.9(c,d) show, for each of the top and bottom conflict regions, the triangles which define it (left figure) and the intersection region with its associated labels (right figure).

Figures 3.10(a–c) show the original individual reconstructions, while Figures 3.10(d–f) show the result after splitting fairly the intersection region between them. Note that if we assign priorities which favor the pelvis, then the faired reconstruction will converge to the original shapes, resulting in the blue tissue bending around the red one. Figure 3.11 shows the effect of different weights on the bottom region of the soft tissue. For clarity, we colored the portions of the original modified faces with cyan and the skeleton faces with magenta. Note also that we did not apply any post-processing step (e.g., smoothing). Figures 3.10(g–i) show the reconstruction of the algorithm in [BV09] (which is identical to the result of [LBD+08] and similar to the result of [BVG11]) before triangulation and smoothness.



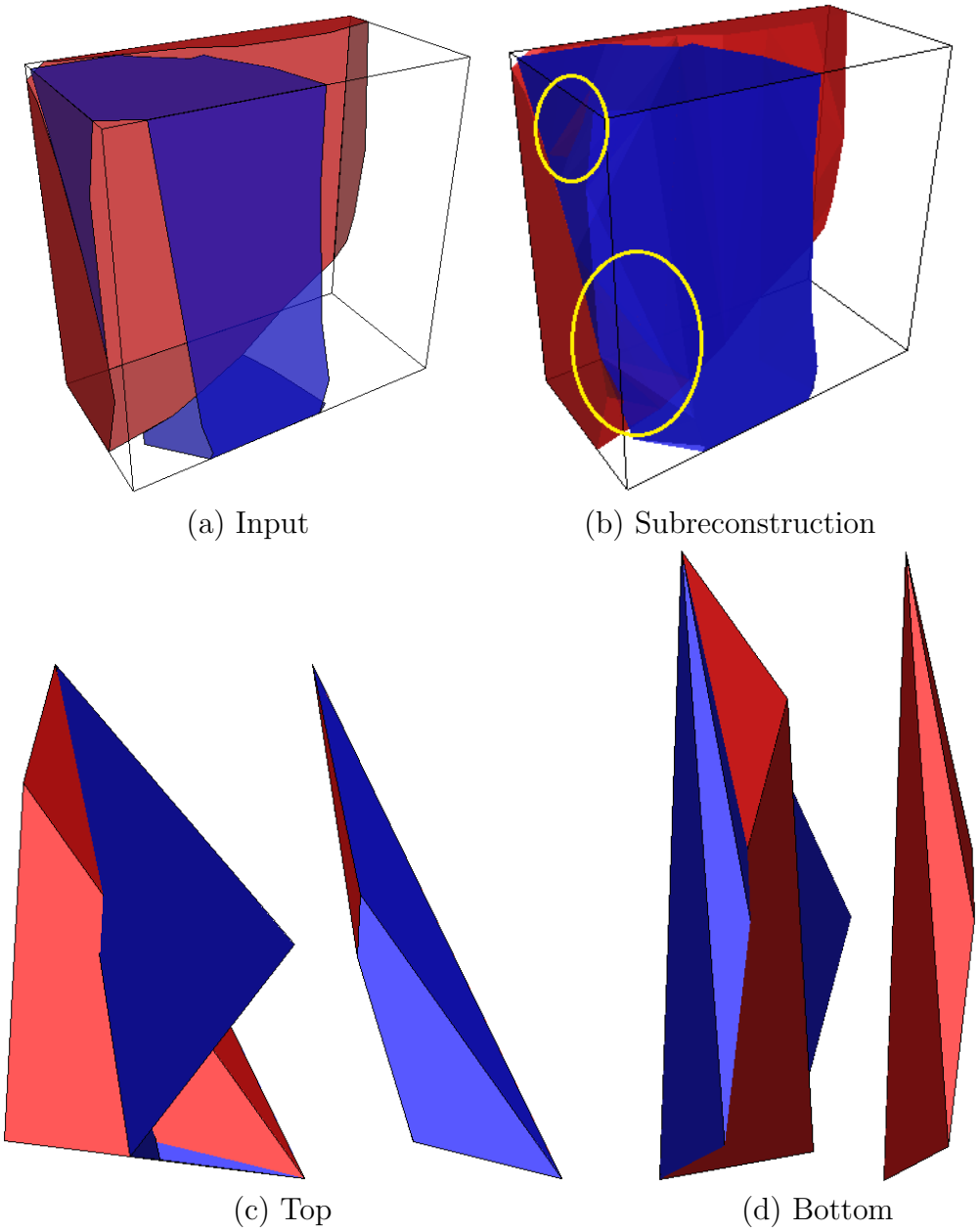
(a) Soft tissue (in blue) curling around part of a pelvis (in red)



(b) Front view (c) Side view

Cross-sections and their labeled contours

FIGURE 3.8: A real example

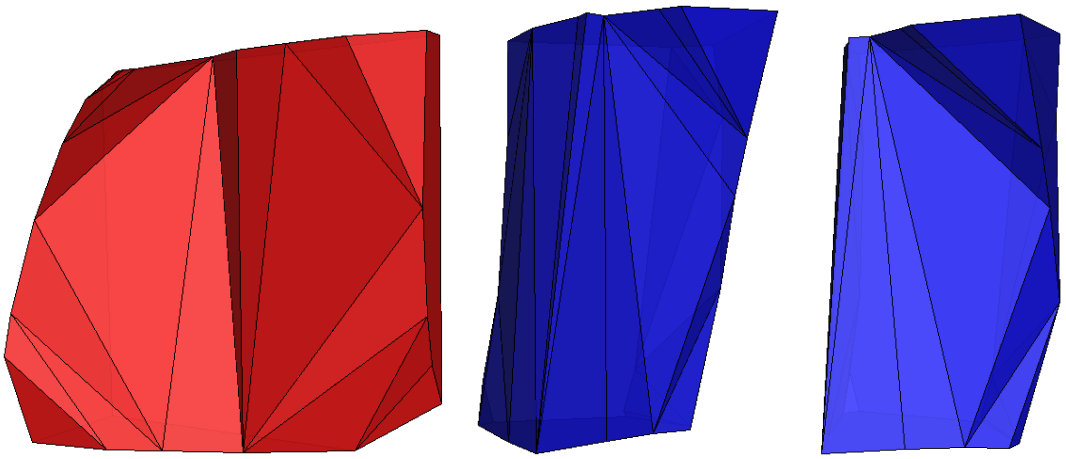


(a) Input (b) Subreconstruction

(c) Top (d) Bottom

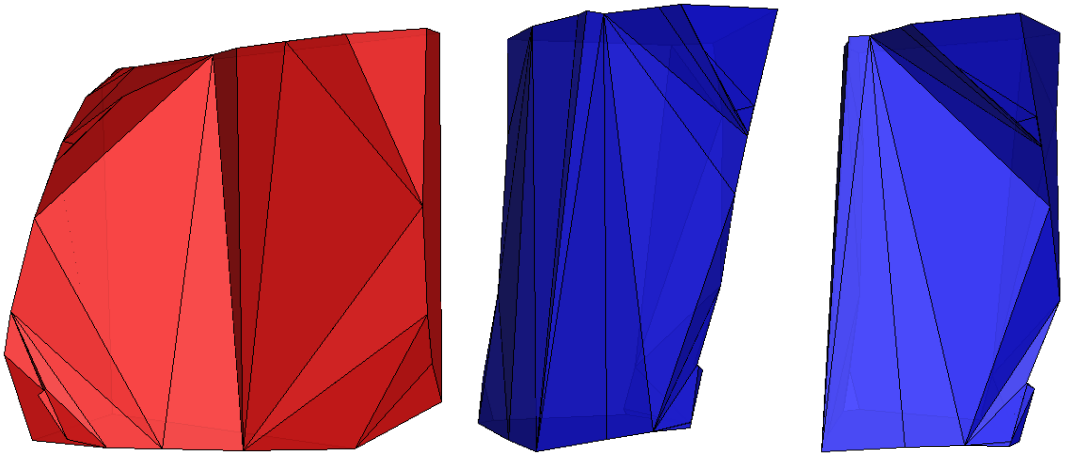
Intersection regions

FIGURE 3.9: Real example: Intersections



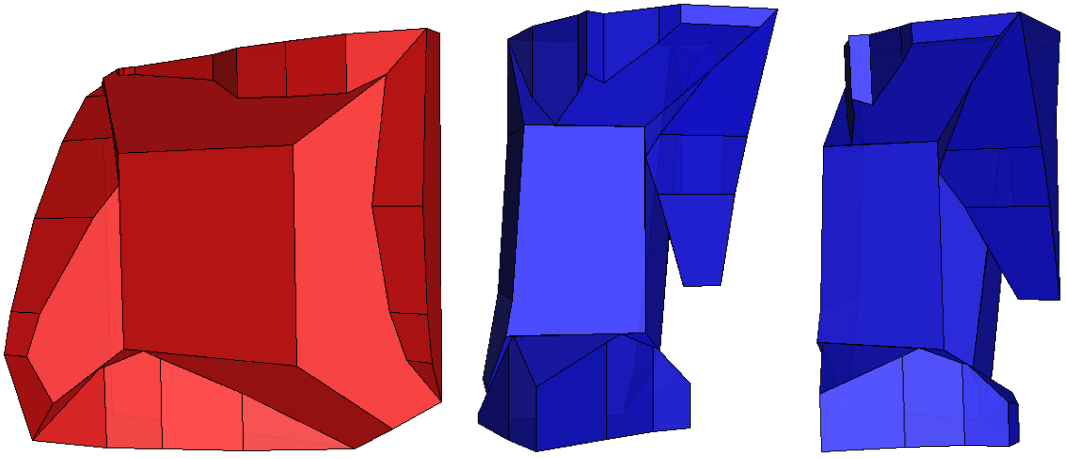
(a) Part of a pelvis (b) Soft tissue (c) Another view

Initial subreconstructions



(d) Part of a pelvis (e) Soft tissue (f) Another view

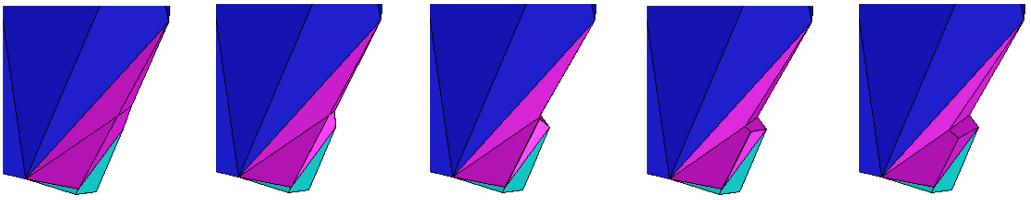
Final subreconstructions



(g) Part of a pelvis (h) Soft tissue (i) Another view

Subreconstructions of [BV09]

FIGURE 3.10: Results



(a) 10:1 (b) 2:1 (c) 1:1 (d) 1:2 (e) 1:10

FIGURE 3.11: The soft tissue (blue) with varying weights (blue:red)

Chapter 4

Conclusion and Future Work

4.1 Summary and Discussion

In this thesis we suggest an algorithm for reconstructing an unknown object from multi-labeled contours. The algorithm reconstructs each type of material separately (using any existing method), then splits the intersection regions in a fair manner.

Although our algorithm is generic and could combine different algorithms for reconstructing each individual material, and although this underlying problem is much easier and potentially could generate a reasonable results, only a few algorithms were suggested. Moreover, they still suffer, in some cases, from bad portions, such as flat regions and thin skinny triangles in their final results.

4.2 Future Work

The main draw back of the algorithm is the computation of the weighted straight skeleton and it's running time complexity. We would like too see how standard collision detection techniques affect the results.

Bibliography

[BCL96] C.L. Bajaj, E.J. Coyle, and K.N. Lin, Arbitrary topology shape reconstruction from planar cross sections. *Graphical Models and Image Processing*, 58 (1996), 524–543.

[BEGV08] G. Barequet, D. Eppstein, M.T. Goodrich, and A. Vaxman, Straight skeletons of three-dimensional polyhedra, *Proc. 16th Ann. European Symp. on Algorithms*, Karlsruhe, Germany, *Lecture Notes in Computer Science*, 5193, Springer-Verlag, 148–160, 2008.

[BGLS04] G. Barequet, M.T. Goodrich, A. Levi-Steiner, and D. Steiner, Contour interpolation by straight skeletons, *Graphical Models*, 66 (2004), 245–260.

[BST00] G. Barequet, D. Shapiro, and A. Tal, Multilevel sensitive reconstruction of polyhedral surfaces from parallel slices, *The Visual Computer*, 16 (2000), 116–133.

[BS96] G. Barequet and M. Sharir, Piecewise-linear interpolation between polygonal slices, *Computer Vision and Image Understanding*, 63 (1996), 251–272.

[BV09] G. Barequet and A. Vaxman, Reconstruction of multi-label domains from partial planar cross-sections, *Computer Graphics Forum*, 28 (2009), 1327–1337.

[BPCC81] S. Batnitzky, H.I. Price, P.N. Cook, L.T. Cook, and S.J. Dwyer III, Three-dimensional computer reconstruction from surface contours for head CT examinations, *J. of Computer Assisted Tomography*, 5 (1981), 60–67.

[BVG11] A. Bermano, A. Vaxman, and C. Gotsman, Online reconstruction of 3D objects from arbitrary cross-sections, *ACM Trans. on Graphics* (electronic), 30 (2011), #113, 14 pp.

[BTS04] A.L. Bogush, A. Tuzikov, and S. Sheynin, 3D object reconstruction from non-parallel cross-sections, *Proc. 17th Ann. IAPR and IEEE Int. Conf. on Pattern Recognition*, Cambridge, England, vol. 3, 542–545, 2004.

[Bo88] J.-D. Boissonnat, Shape reconstruction from planar cross sections, *Computer Vision, Graphics and Image Processing*, 44 (1988), 1–29.

[BG92] J.-D. Boissonnat and B. Geiger, Three dimensional reconstruction of complex shapes based on the Delaunay triangulation, Technical Report 1697, Inria-Sophia Antipolis, 1992.

[BM07] J.-D. Boissonnat and P. Memari, Shape Reconstruction from unorganized cross-sections, *Proc. Symp. on Geometry Processing*, Barcelona, Spain, 89–98, 2007.

[CS78] H.N. Christiansen and T.W. Sederberg, Conversion of complex contour line definitions into polygonal element mosaics, *Computer Graphics*, 13 (1978), 187–192.

[CP94] Y.K. Choi and K.H. Park, A heuristic triangulation algorithm for multiple planar contours using an extended double branching procedure. *The Visual Computer*, 10 (1994), 372–387.

[CLLC88] H.E. Cline, W.E. Lorensen, S. Ludke, C.R. Crawford, and B.C. Teeter, Two algorithms for the three-dimensional reconstruction of tomograms, *Medical Physics*, 15 (1988), 320–327.

[CGAL] Computational Geometry Algorithms Library. *http://www.cgal.org*

[CCLB80] L.T. Cook, P.N. Cook, K.R. Lee, S. Batnitzky, B.Y.S. Wong, S.L. Fritz, J. Ophir, S.J. Dwyer III, L.R. Bigongiari, and A.W. Templeton, An algorithm for volume estimation based on polyhedral approximation, *IEEE Trans. on Biomedical Engineering*, 27 (1980), 493–500.

[DP97] C.R. Dance and R.W. Prager, Delaunay reconstruction from multiaxial planar cross-sections, Technical Report CUED/F-INFENG/TR273, Cambridge Univ., England, 1997.

[EB11] J. Edwards and C. Bajaj, Topologically correct reconstruction of tortuous contour forests, *Computer-Aided Design*, 43 (2011), 1296–1306.

[EPO91] A.B. Ekoule, F.C. Peyrin, and C.L. Odet, A triangulation algorithm from arbitrary shaped multiple planar contours, *ACM Trans. on Graphics*, 10 (1991), 182–199.

[FKU77] H. Fuchs, Z.M. Kedem, and S.P. Uselton, Optimal surface reconstruction from planar contours, *Communications of the ACM*, 20 (1977), 693–702.

[GD82] S. Ganapathy and T.G. Dennehy, A new general triangulation method for planar contours, *ACM Trans. on Computer Graphics*, 16 (1982), 69–75.

[Ge93] B. Geiger, Construction et utilisation des modèles d'organes en vue de l'assistance au diagnostic et aux interventions chirurgicals, Ph.D. Dissertation, L'Ecole des Mines de Paris, 1993.

[JWC+05] T. Ju, J.D. Warren, J. Carson, G. Eichele, C. Thaller, W. Chiu, M. Bello, and I.A. Kakadiaris, Building 3D surface networks from 2D curve networks with application to anatomical modeling, *The Visual Computer*, 21 (2005), 764–773.

[KD88] N. Kehtarnavaz and R.J.P. De Figueiredo, A framework for surface reconstruction from 3D contours, *Computer Vision, Graphics and Image Processing*, 42 (1988), 32–47.

[KSD88] N. Kehtarnavaz, L.R. Simar, and R.J.P. De Figueiredo, A syntactic/semantic technique for surface reconstruction from cross-sectional contours, *Computer Vision, Graphics and Image Processing*, 42 (1988), 399–409.

[Ke75] E. Keppel, Approximating complex surfaces by triangulation of contour lines, *IBM J. of Research and Development*, 19 (1975), 2–11.

[LBD+08] L. Liu, C. Bajaj, J. Deasy, D.A. Low, and T. Ju, Surface reconstruction from non-parallel curve networks, *Computer Graphics Forum*, 27 (2008), 155–163.

[LC87] W.E. Lorensen and H.E. Cline, Marching cubes: A high resolution 3D surface construction algorithm, *Computer Graphics*, 21 (1987), 163–169.

[MSS91] D. Meyers, S. Skinner, and K. Sloan, Surfaces from contours: The correspondence and branching problems, *Proc. Graphics Interface*, Calgary, Alberta, Canada, 246–254, July 1991.

[OPC96] J.-M. Oliva, M. Perrin, and S. Coquillart, 3D reconstruction of complex polyhedral shapes from contours using a simplified generalized Voronoi diagram, *Computer Graphics Forum*, 15 (1996), C397–408.

[PT94] B.A. Payne and A.W. Toga, Surface reconstruction by multiaxial triangulation, *IEEE Computer Graphics Applications*, 14 (1994), 28–35.

[Sh81] M. Shantz, Surface definition for branching contour-defined objects, *Computer Graphics*, 15 (1981), 242–270.

[SH81] K.R. Sloan and L.M. Hrechanyk, Surface reconstruction from sparse data, *Proc. IEEE Conf. on Pattern Recognition and Image Processing*, Dallas, TX, 1981, 45–48.

[SP88] K.R. Sloan and J. Painter, Pessimal guesses may be optimal: A counterintuitive search result, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10 (1988), 949–955.

[SSBT01] T. Surazhsky, V. Surazhsky, G. Barequet, and A. Tal, Blending polygonal shapes with different topologies, *Computers and Graphics*, 25 (2001), 29–39.

[WA86] Y.F. Wang and J.K. Aggarwal, Surface reconstruction and representation of 3-D scenes, *Pattern Recognition*, 19 (1986), 197–207.

[WW93] E. Welzl and B. Wolfers, Surface reconstruction between simple polygons via angle criteria, *Proc. 1st Ann. European Symp. on Algorithms*, *Lecture Notes in Computer Science*, 726, Springer Verlag, 1993, 397–408.

[ZJH87] M.J. Zyda, A.R. Jones, and P.G. Hogan, Surface construction from planar contours, *Computers and Graphics*, 11 (1987), 393–408.

בצעד 3 אנחנו מחשבים את השלד התלת מימדי המשוקלל, כאשר משקלי הפאות הם אלה שנקבעו בצעד הקודם.

בצעד 4 אנחנו מציבים שתי תוויות לכל פאת שלד שנוצרה בצעד הקודם. כל פאת שלד שייכת לשני תאי שלד בשתי אוריינטציות שונות, ולכן היא מקבלת את שתי התוויות של התאים האלו. יש להזכיר שכל תא שלד נפרש ע"י פאת חיתוך אחת ולכן הוא מקבל את התווית שלה. לבסוף, אנחנו בוחרים פאות שלד גבוליות, כלומר פאות שקיבלו שתי תוויות שונות, ומשלשים אותן. פאות אלו משוכפלות ומשוייכות לשני החומרים המתחרים בהתאם לשתי התוויות שקיבלה כל פאה.

הפלט הסופי של האלגוריתם הוא אוסף המשולשים שהתקבלו מצעד 1 ו-4.

איורים 2.1 – 2.6 מבהירים את צעדי האלגוריתם השונים. באיור 2.7 אנו מציגים דוגמה להמחשת ההשפעה של המשקלים השונים והצורות השונות של החיתוך על התוצאה הסופית. שילוב שני הנתונים האלו מביא לתוצאות הגיוניות וטבעיות.

במשפט 2.1 אנו טוענים כי פאות השלד שנבחרו להשתתף בתוצאה הסופית מתחברות היטב עם הפאות המתחרות.

איורים 3.5 – 3.7 מציגים דוגמה סינתטית, ומשווים בין תוצאות ההרצה של האלגוריתם שלנו מול אלו של האלגוריתמים הקיימים. דוגמה זו מציגה בנוסף בפירוט רב יותר את צעדי האלגוריתם ומבהירה את שיטת העבודה.

איורים 3.8 – 3.11 מציגים טיפול בתא שנלקח מדוגמה אמיתית ומשווים את התוצאות שלנו עם תוצאות הרצה של האלגוריתמים הקיימים. בנוסף, הם מציגים את השפעת התערבות המשתמש (ע"י קביעת מערכת הכללים) על התוצאה הסופית, כך שאנחנו מסיקים לבסוף שהתערבות טבעית/הגיונית תורמת ליצירת שחזורים טבעיים/הגיוניים הקרובים מאוד לאובייקט המקורי שנחקר.

עבודה זו מורכבת מארבעה פרקים. בפרק הראשון אנו מציגים את הרקע לבעיה, פותחים במבוא ואז מגדירים את הבעיה באופן מדויק. בפרק השני אנו מציגים את האלגוריתם, תחילה את הרעיון המרכזי ואז את השלבים בפירוט רב יותר, ונעזרים בדוגמה להמחשה. בפרק השלישי אנו מציגים תחילה את המימוש ואז מראים תוצאות הרצה של האלגוריתם על דוגמה סינתטית ועל דוגמה אמיתית, ומשווים את התוצאות שלנו עם התוצאות המתקבלות מהרצת אלגוריתמים קיימים. בפרק הרביעי והאחרון אנו מסכמים את העבודה ומציגים דיון קצר.

האלגוריתם, כצעד ראשון, משלים את החתכים בדומה ל- [BVG11] ובונה את חלוקת המרחב התלת-מימדי. לאחר מכן הוא משחזר כל חומר בנפרד ע"י הפעלת אחד האלגוריתמים הקיימים (למשל: [DP97, BTS04, BM07]). ולבסוף, הוא מגלה את החיתוכים בכל תא של חלוקת המרחב ומטפל בהם. המשתמש יכול להתערב בטיפול בחיתוכים. למטרה זו אנחנו מציעים מערכת כללים. המשתמש יכול לקבוע אותה מראש או באופן אינטרקטיבי תוך כדי הטיפול בחיתוכים. מערכת כזו מכילה את הכללים הבאים:

1. חומר מסוג מסויים יכול להישאר כמו שהוא. כלומר הוא "מנצח".
2. חומר שמהווה צינור החותך חומר אחר יכול להישאר כמו שהוא (ללא תלות בסוג החומר).
3. ניתן לתת עדיפויות לחומרים השונים (כלומר, לדרג אותם) כדי לפתור התנגשויות בין שניים או יותר חומרים אשר נבחרו להישאר כמו שהם.
4. ניתן לתת משקלים לחומרים השונים על מנת לחלק את אזור החיתוך בין המתחרים בצורה הוגנת. ברירת המחדל היא משקל שווה לכולם.

כל החוקים הם גיאומטרים מטבעם, ומטרתם היא לחלק את אזור החיתוך בין המתחרים. לחלק מהחוקים יש בנוסף טעם טופולוגי. למשל, כאשר כלי דם עובר דרך שריר, כלל 2 יאפשר לא לשבור אותו לשני חלקים לא קשירים. בדוגמה אחרת, עצם קשה יותר משריר, לכן, מתן משקל גבוה לטובת העצם ע"י כלל 4 יכול לגרום לשריר להתעקל סביב העצם אשר תשמור על צורתה המקורית. בנוסף, נוכל להגדיר כלים גלובאליים במובן של שיתוף מידע בין תאים שונים בחלוקת המרחב. למשל, נוכל לתת משקל נמוך לחומר המעורב במספר רב של חיתוכים או לשחזר אותו ע"י אלגוריתם אחר.

לאחר שהמשתמש קובע את דרישותיו, האלגוריתם מתחיל לצמצם את אזורי החיתוך בעזרת כללים 1-3. לאחר מכן, לכל איזור חיתוך קשיר שנותר ללא טיפול:

1. מחסרים אותו מהשחזורים אשר גרמו לחיתוך להווצר, להם נקרא השחזורים המתחרים.
2. מציבים תווית ומשקל לכל פאה.
3. מחשבים את השלד התלת מימדי המשוקלל של איזור החיתוך.
4. קובעים שתי תוויות לכל פאת שלד שנוצרה, ובוחרים את הפאות עם שתי תוויות שונות.

להלן הסברים לשלבים הנ"ל.

בצעד 1 אנו מחסרים את אזור החיתוך מהשחזורים המתחרים ומורידים את הפאות (או חלקי פאות) שלו מהם. לאחר מכן אנו משלשים את הפאות שנוצרו.

בצעד 2 כל פאה של אזור החיתוך מקבלת תווית לפי השחזור המתחרה המכיל אותה ונותנים לה משקל השווה למשקל של אותה תווית (סוג של חומר).

תקציר

בעבודה זו אנחנו מציעים אלגוריתם הוגן לשחזור אובייקט רב-תוויות מחתכים שלו. המניע העיקרי של בעיה זו בא מתחום ההדמיה הרפואית, כשהחתכים של איברי הגוף השונים של בן-אדם התקבלו מצילומי CT או MRI.

בעיה זו משכה את תשומת ליבם של חוקרים במשך 35 השנים האחרונות. הבעיה מוגדרת כך: בהינתן אוסף של חתכים, כל חתך מורכב מאוסף של פוליגונים סגורים, פשוטים ולא נחתכים, המטרה היא ליצור משטח סגור העובר בחלל התחום בין החתכים השונים ומסכים עם הפוליגונים הנתונים בקלט.

רוב העבודות הקודמות הניחו שהחתכים מקביליים ושכל חתך מכיל פוליגון אחד. לאחר מכן טיפלו בבעיה הכללית יותר בה לא היתה הנחה כלשהי על מספר הפוליגונים ולא על רמת הקינון שלהם. הרחבות נוספות לבעיה היו כדלקמן: החתכים לא חייבים להיות מקביליים ולא חייבים להיות שלמים, כלומר ניתן היה לקבל חתך עם נתונים חלקיים בלבד. הרחבה זו נבעה משיקולי מניעה של הקרנת יתר של המטופלים. הרחבה נוספת ואחרונה לבעיה זו אופיינה בהשמת תוויות לפוליגונים השונים, כאשר תוויות שונות מתארות פוליגונים המתאימים לחומרים מסוגים שונים. חומרים כאלו יכולים לתאר, למשל, עצמות, כלי דם ושרירים.

בעבודה זו אנחנו מציעים אלגוריתם אשר פותר את הבעיה הכללית ביותר. כלומר, החתכים נתונים באוריינטציה כלשהי והם לאו דווקא שלמים, ודוגמים אובייקט רב-תוויות. קיימים רק שני אלגוריתמים המטפלים בבעיה הכללית הזו [BV09, BVG11] ולהם יש פלט דומה. ההבדל ביניהם הוא ש- [BVG11] מקוון, כלומר אפשר להכניס אליו עוד חתכים כקלט תוך כדי החישוב. קיים עוד אלגוריתם הפותר את הבעיה הפחות כללית בה מניחים שהחתכים שלמים [LBD+08], והוא מקרה פרטי של האלגוריתם [BV09] ולכן יש לו פלט זהה. בשלוש העבודות האלו מוצעת שיטת אינטרפולציה המשחזרת את כל החומרים יחד, ובנוסף הם מתיחסים לאוויר (הרווח בין הפוליגונים השונים בחתכים) כאל חומר רגיל במטרה להימנע מיצירת חיתוכים בין השחזורים השונים. לעומת זאת, אנחנו מציעים גישה אחרת לטיפול בבעיה בה כל חומר משוחזר בנפרד, ואם נוצרים חיתוכים בין השחזורים השונים, שחזורים אלו חולקים ביניהם את אזור החיתוך בצורה הוגנת. שחזור כל חומר בנפרד הוא בעיה הרבה יותר פשוטה עם פחות אילוצים ויותר גמישות, תכונות העוזרות ליצור שחזורים נעימים יותר לעין ובאופן פשוט יותר. מצד שני, תוצאות הניסויים שערכנו מראות שבשחזור מחתכים הדוגמים אובייקטים אמיתיים מתקבלים חיתוכים אשר נוצרו מחפיפות בגבולות של חומרים/שחזורים שונים, כך שאחרי שמטפלים בהם (מגיעים לפשרה ומחלקים אותם לחומרים המתחרים) התוצאה המתקבלת נראית נעימה לעין. לאורך עבודה זו אנו מציגים מספר דוגמאות המשוות בין שתי הגישות האלו.

המחקר נעשה בהנחיית פרופ' גיל ברקת

בפקולטה למדעי המחשב

הכרת תודה

אני מודה לטכניון ולמשרד המדע והטכנולוגיה על התמיכה הכספית הנדיבה במשך השתלמותי.

שחזור רב-תוויות הוגן מחתכים

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר מגיסטר למדעים
במדעי המחשב

ראידה נעאמנה

הוגש לסנט הטכניון – מכון טכנולוגי לישראל

תשרי תשע״ד חיפה ספטמבר 2013